تعریف چند سروری و چند دیتابیسی :

ارتباط بین جدول Product از دیتابیس محصولات (DBProduct) و جدول ProducItem از دیتابیس خدمات (DBProductService ) که نه تنها از پایگاه داده مختلف می باشند بلکه به روی سرور های مختلفی قرار دارند. به این ترتیب امکان استفاده از این رابطه در ورود اطلاعات هر طرف ممکن است. همچنین امکان استفاده از این ارتباط در جستجو و نمایش اطلاعات میسر می باشد. برای مثال ستون نوع محصول که از جدول ProductType در DBProducts می باشد در لیست نمایش پیش فرض ProducItem تعریف شده است که البته به علت بین سروری بودن این ارتباط تعریف لینک سرور در پایگاه داده مورد نیاز است.

مشخصات موجودیتها :

امکان بارگذاری خودکار داده ها : جداول پایه مانند برند و نوع گارانتی

تولید خودکار لیست نمایش پیش فرض :

قانون : 1- ستونهای کلید اصلی موجودیت اضافه میشوند 2- ستونهای ساده غیر کلید اصلی که کلید فرعی نیز نیستند بر اساس 3 مرحله زیر شناسایی و اضافه میشوند. مراحل : ....

3- ستونهای روابط اضافه میشوند (شرح جزئیات)

تولید خودکار لیست جستجوی پیش فرض

قانون : 1- ستونهای کلید اصلی موجودیت اضافه میشوند 2- ستونهای ساده غیر کلید اصلی که کلید فرعی نیز نیستند بر اساس 3 مرحله زیر شناسایی و اضافه میشوند. مراحل : ....

3- ستونهای روابط اضافه میشوند (شرح جزئیات)

بصورت کلی تمامی فرمهای استخراج اولیه نیاز به تشریح دارند

# امکان تعریف موجودیت ها:

موجودیتها در مدل فراداده، از پایگاه داده هدف استخراج می شوند. هر موجودیت می تواند عیناً معادل یک جدول در پایگاه داده هدف باشد و یا انشعابی از یک جدول. فارق از ساختار اصلی موجودیت ها که شامل خصوصیات، روابط و ... می باشد، اکثر بخشهای مدل فراداده بر محوریت موجودیت ها تعریف می شوند که در جای خود به آنها و ارتباطشان با موجودیتها اشاره خواهد شد. امّا خود موجودیت در مدل فراداده دارای خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Name | نام موجودیت (عموماً نام انگلیسی جدول معادل) |
| Alias | عنوان انتخابی موجودیت |
| IsDisabled | غیرفعال بودن موجودیت |
| IsReadonly | فقط خواندی بودن موجودیت |
| Removed | تعیین کننده اینکه آیا موجودیت به کل حذف شده است |
| BatchDataEntry | ورود اطلاعات چندتایی موجودیت بصورت یکجا |
| Color | رنگ پیش فرض موجودیت در نمودارها |
| SearchInitially | تعیین کننده اینکه آیا داده های موجودیت در فرمهای جستجو بصورت اولیه جستجو شوند و دریافت شوند |
| EntityListViewID | لیست نمایش پیش فرض برای موجودیت |
| EntitySearchID | لیست جستجوی پیش فرض برای موجودیت |
| Description | توضیحات مربوطه برای موجودیت |
| IsOrginal | تعیین اینکه موجودیت عیناً دارای جدول در پایگاه داده هدف است یا مشتق شده از موجودیت اصلی دیگری است. |
| LoadArchiveRelatedItems | تعیین اینکه موارد آرشیو مرتبط با موجودیت بصورت پیش فرض نمایش داده شوند یا خیر |
| LoadLetterRelatedItems | تعیین اینکه نامه های مرتبط با موجودیت بصورت پیش فرض نمایش داده شوند یا خیر |
| IsView |  |
|  |  |

در ادامه به برخی کاربردهای خصوصیات کلیدی موجودیت ها پرداخته می شود.

## کاربرد خصوصیات موجودیتها در فرم های ورود اطلاعات :

### خصوصیت Removed :

|  |
| --- |
| توضیحات |
| تعیین حذف شده بودن موجودیت از پایگاه داده هدف. این خصوصیت تنها از پایگاه داده هدف و در صورت حذف شدن موجودیت از پایگاه داده هدف قابل مقداردهی می باشد (BizTableDrivedEntity.UpdateModelFromTargetDB: 1bcad93c-9349-4f31-a710-b2bc55f8b578). در صورت trueبودن این خصوصیت، موجودیت در هیچ جای برنامه دیگر قابل دسترس نمی باشد(BizTableDrivedEntity. GetAllEntities: fc78bf47-f1ed-4bee-b141-8b4f7f33ae0d). |

### خصوصیت IsDisabled :

|  |
| --- |
| توضیحات |
| تعیین فعال بودن موجودیت. این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است (BizTableDrivedEntity.UpdateFromUI: 03385f9c-7b5c-431f-b532-fd4a517eeb6e) . |
| در خواندن رابطه های مدل فراداده نیز بسته به اینکه موجودیتهای فعال مورد نیازند یا تمامی موجودیت ها از این خصوصیت برای فیلتر کردن استفاده می شود(BizTableDrivedEntity.GetAllEnabledEntities: 53e163cd-40df-4694-8412-05df4ec41a23). |
| موجودیت های مشتق شده که دیگر نیازی به آنها نداریم را بجای پاک کردن غیر فعال می کنیم ( BizTableDrivedEntity.RemoveDrivedEntity: 53e163cd-40df-4694-8412-05df4ec41a23) |

### خصوصیت IsReadonly :

|  |
| --- |
| توضیحات |
| تعیین فقط خواندنی بودن موجودیت. این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است (BizTableDrivedEntity.UpdateFromUI: 03385f9c-7b5c-431f-b532-fd4a517eeb6e) . |
| فقط خواندنی بودن موجودیت در فرم ورود اطلاعات کنترل می شود. به عبارتی اگر موجودیت فقط خواندنی باشد، تمامی ستونها و روابط فرعی به اصلی فقط خواندنی می شوند. (BizTableDrivedEntity.RemoveDrivedEntity: 224c5e70-1ba3-43d1-a47b-a8c7523b7591 و 46239ff9-cd78-4831-ad4a-8be394c313b9) |
| کنترل دسترسی حذف داده. (BizTableDrivedEntity. DataHasNotDeleteAccess: a34e3c07-b7ca-4b78-9b00-f7b87aadf83b و 46239ff9-cd78-4831-ad4a-8be394c313b) |

### خصوصیت BatchDataEntry :

|  |
| --- |
| توضیحات |
| * تعین اینکه فرم ورود اطلاعات موجودیت چندتایی باز شود (مانند گرید) و یا ورود اطلاعات تک به تک انجام شود (BaseEditEntityArea.GetEditEntityArea: 1a7810b0-c6dd-4c12-8ad6-9ad46f1525db). این خصوصیت توسط کاربر در برنامه مدیریت فراداده تعیین می گردد. |

### خصوصیت Color :

|  |
| --- |
| توضیحات |
| این خصوصیت توسط کاربر در برنامه مدیریت فراداده تعیین می گردد و تعیین کننده رنگ معادل هر موجودیت در نمودارها و نمایش های گرافیکی است |
| کاربرد در نمایش آیتمهای آرشیو موجودیت (847567e9-a0da-44fb-81d8-713ac4c4f6e8) |

### خصوصیت SearchInitially :

|  |
| --- |
| توضیحات |
| تعیین اینکه جستجوی داده های موجودیت در فرمهای ورود اطلاعات بصورت خودکار انجام شود(b07a3760-ff93-410f-b6b9-e40244e72ac7). |

### خصوصیت IsView :

|  |
| --- |
| توضیحات |
| تعیین اینکه موجودیت معادل در پایگاه داده هدف، یک نما است یا خیر (یک جدول است) |
| فیلتر کردن درخت اشیاء مدل فراداده (BizDatabaseToObject.GetDatabaseChildObjects: a5eceb3c-172f-4a51-91be-c1e9819478c9) |
| فیلتر کردن دریافت لیست موجودیتها (BizTableDrivedEntity.GetAllEntitiesExceptViews: 1f144c7d-c4ba-4c79-a605-1ab97a637518) |
| این خصوصیت از موجودیتهای پایگاه داده مقصد بروزرسانی می شود (BizTableDrivedEntity.UpdateEntityFromTargetDB: 86b1c396-2004-4535-8de2-f4a43b5f5d62) |
| برای نماها نحوه ایجاد Query ها متفاوت است (SearchRequestProcessor.GetFromQuery: 72ca04ba-004f-45aa-afb4-49a7ec0e684b) |
| برای نماها نحوه دریافت تنظیمات منوها متفاوت است (BizDataMenuSetting.GetDataMenu: 1677d2a6-f3cf-43f9-b61b-f6bf4c34c203 و71a75126-c643-4bbd-b001-8605f9c20cc9 ) |
| برای نماها نحوه تولید لیست نمایش پیش فرض متفاوت است (BizEntityListView.GenereateDefaultListViewColumns: 35838f12-76ec-4a3a-8f4d-e0b50248b309) |
| برای نماها نحوه تولید لیست جستجوی پیش فرض متفاوت است (BizEntitySearch.GenereateDefaultSearchColumns: 8ab306e9-0d52-4be6-95c0-9e5b4c36a21c) |
| برای نماها امکانات موجودیت ها در برنامه مدیریت فراداده متفاوت است (d87f4811-14dd-49ce-b86f-087423a635d6) |
| برای نماها نحوه تعریف لیست نمایش در برنامه مدیریت فراداده متفاوت است (1db5ddd3-be11-41b3-9739-a97eb7d1f22e) |
| برای نماها نحوه تعریف رابطه در برنامه مدیریت فراداده متفاوت است (ea5c5926-4277-43d7-a4da-7c2af6a7543d) |
| برای نحوه تعریف لیست جستجو در برنامه مدیریت فراداده متفاوت است (4bbbbc10-0d57-4023-821b-62576df5955b و 689066db-d730-43a0-b5c7-9eddceb0a41b) |

# امکان تعریف انواع ستونها:

ستونهای جداول که در مدل فراداده همان خصوصیات موجودیتها می باشند، از پایگاه داده هدف استخراج می شوند. هر ستون دارای خصوصیات عمومی زیر می باشد.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Name | نام خصوصیت (عموماً نام انگلیسی ستون معادل) |
| Alias | نام فارسی معادل |
| IsDisabled | غیرفعال بودن ستون |
| IsReadonly | فقط خواندنی بودن ستون |
| DataType | نوع داده ستون |
| IsNull | Null پذیر بودن ستون |
| PrimaryKey | آیا کلید اصلی است؟ |
| ForeignKey | آیا کلید فرعی است؟ |
| DataEntryEnabled | آیا ستون قابل ورود اطلاعات است |
| DefaultValue | مقدار پیش فرض |
| IsMandatory | اجباری بودن داشتن مقدار |
| IsIdentity | آیا ستون دارای شمارنده خودکار است |
| DBCalculateFormula | فرمول محاسبه مقدار ستون در پایگاه داده |

## کاربرد خصوصیات ستونها در فرم های ورود اطلاعات :

### خصوصیت IsMandatory:

|  |
| --- |
| توضیحات |
| تشخیص اجباری بودن خصوصیت نیز برای اولین بار بصورت خودکار از Null پذیر بودن ستون انجام می شود (BizColumn.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff) |
| این خصوصیت توسط کاربر در برنامه مدیریت فراداده نیز قابل تعیین است (BizColumn.UpdateColumnsFromUI: 8ee14cf5-30d9-4af4-afc3-14ecf5010471) |
| افزودن ستاره به عناوین خصوصیت های اجباری (BaseEditEntityArea.GenerateControls: 8218f45f-1045-4f73-8fe6-0a7df332e77f) |
| تولید تولید رنگ و تولتیپ برای خصوصیت (ChildSimpleContorlProperty.SetControlUIDetails: ffee3df3-b262-4bb8-88ba-2b095307e547) |
| کنترل خصوصیات اجباری امّا غایب (UIValidationManager.CheckAccessValidation: 511a49ba-1a7c-47d0-8b6f-3b7eb162083e) |
| کنترل اجباری بودن ورود اطلاعات خصوصیت (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت Removed :

|  |
| --- |
| توضیحات |
| تعیین حذف شده بودن ستون از پایگاه داده هدف. این خصوصیت تنها از پایگاه داده هدف و در صورت حذف شدن ستون از پایگاه داده هدف قابل مقداردهی می باشد (BizColumn.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff). در صورت trueبودن این خصوصیت، ستون در هیچ جای برنامه دیگر قابل دسترس نمی باشد(835426f1-378b-41df-af1a-76435880babc)، همچنین روابطی که روی آن ستون برقرار باشند (e5f2c46e-fe5d-4f6d-b0ba-8a8242e6eb63) |

### خصوصیت IsDisabled :

|  |
| --- |
| توضیحات |
| تعیین فعال بودن ستون. این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است (BizColumn.UpdateColumnsFromUI: 8ee14cf5-30d9-4af4-afc3-14ecf5010471) . |
| در خواندن ستونهای مدل فراداده نیز بسته به اینکه ستونهای فعال مورد نیازند یا تمامی ستونها از این خصوصیت برای فیلتر کردن استفاده می شود(dc04b084-bb2d-4e9a-9710-b0943ebf3267). |
| در خواندن روابط هم به فعال بودن یا نبودن ستون های رابطه توجه می شود (BizRelationship.GetAllEnabledRelationships: 32c5d8c8-dccc-4436-b41b-ba141918b2bb و BizRelationship.ToRelationshipDB: 3424ff73-1465-4872-aa31-c2cecbba5f18) |

### خصوصیت DataEntryEnabled :

|  |
| --- |
| توضیحات |
| این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است (BizColumn.UpdateColumnsFromUI: 8ee14cf5-30d9-4af4-afc3-14ecf5010471) |
| تعیین نمایش خصوصیت در فرم ورود اطلاعات(BizTableDrivedEntity.CheckDataEntryPermission: 46239ff9-cd78-4831-ad4a-8be394c313b9). |
| به صورت کلی در کلاسهای تولید پیش فرض ترکیب واسط کاربری (BizEntityUIComposition.GetOrCreateEntityUIComposition: 3adc6e16-aff9-4bc6-a425-33f45647385a) |
| تعیین قابل ورود اطلاعات بودن رابطه خارجی به اصلی به روی ستونهای کلید خارجی نگه داری می شوند (2be49890-963b-4d7e-ace1-f30042893bc1) |
| کنترل خصوصیات اجباری امّا غایب (UIValidationManager.CheckAccessValidation: 511a49ba-1a7c-47d0-8b6f-3b7eb162083e) |

### خصوصیت IsReadonly :

|  |
| --- |
| توضیحات |
| تعیین فقط خواندنی بودن خصوصیات. این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است(BizColumn.UpdateColumnsFromUI: 8ee14cf5-30d9-4af4-afc3-14ecf5010471) |
| تعیین قابل ورود اطلاعات بودن روابط به روی ستونهای کلید خارجی نگه داری می شوند (BizRelationship.ToRelationshipDTO: 2be49890-963b-4d7e-ace1-f30042893bc1 f30042893bc1 و BizRelationship.ToRelationshipDB: 3424ff73-1465-4872-aa31-c2cecbba5f18) |
| فقط خواندنی بودن خصوصیات در دریافت موجودیت ورود اطلاعات کنترل می شوند (BizTableDrivedEntity.CheckDataEntryPermission: 46239ff9-cd78-4831-ad4a-8be394c313b9). |
| در فرمهای ورود اطلاعات، تعیین فقط خواندنی بودن خصوصیت داده (ChildSimpleContorlProperty.IsReadonly: 7a082dc6-eb74-4622-8062-1eac557339df) |
| تولید تولید رنگ و تولتیپ برای خصوصیت (ChildSimpleContorlProperty.SetControlUIDetails: ffee3df3-b262-4bb8-88ba-2b095307e547) |

### خصوصیت DefaultValue :

|  |
| --- |
| توضیحات |
| تعیین مقدار پیش فرض خصوصیات. این خصوصیت از پایگاه داده هدف بروزرسانی می شود(UpdateColumnsFromTargetDB.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff). |
| در فرمهای ورود اطلاعات، به هنگام ایجاد داده جدید مقدار پیش فرض خصوصیات نمایش داده می شود(AgentHelper.CreateAreaInitializerNewData: c8c4ca7a-9a7c-4469-a872-193cdabb907a). |
| در فرمهای ورود اطلاعات، تعیین اینکه مقدار خصوصیت تغییری کرده است یا خیر (EntityInstanceProperty.ValueIsEmptyOrDefaultValue: 9b38140b-d290-4157-a617-b1b659034738) |

### خصوصیت IsIdentity:

|  |
| --- |
| توضیحات |
| تعیین اینکه ستون در پایگاه داده هدف شماره انداز خودکار دارد. این خصوصیت از پایگاه داده هدف بروزرسانی می شود(UpdateColumnsFromTargetDB.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff). |
| برگرداندن شماره جدید ستون بعد از اجرای دستور Insert برای بروزرسانی کلیدهای خارجی وابسته (ConnectionManager.ExecuteTransactionalQueryItems: 74d5cc97-f11a-4579-b922-9475f2414cf3). |
| کاربرد در ساخت وابستگی های قلم داده ها در کلاسهای نهایی بروزرسانی داده (EditQueryItemManager.SetSortedTree: b6bbadec-6c6a-4b8c-a604-9703604e81b5 و EditQueryItemManager.GetUpdateQueryItems: be3c95de-74cf-4838-aa97-c6809a1b6f29) |
| در فرمهای ورود اطلاعات خواندن کلید ایجاد شده بعد از ثبت داده جدید (EditRequestManager.Process: adb34385-d2d1-4b29-a147-0cbaba268135) |
| استفاده در ایجاد لاگ داده(EntityAreaLogManager.ToDataLog: 131e42d5-ee2e-4613-a5d6-b7c6b5787207)  چالش : لاگ داده نهایتاً با شناسه جدید ثبت می شود یا فقط با عبارت "<identity>"؟ |
| در فرم های ورود اطلاعات، اعتبارسنجی ورود اطلاعات شدن اجباری فیلد (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |
| تعیین اینکه ستون یک خصوصیت ساده است یا خیر (ColumnDTO.IsSimpleColumn: edeace71-ffce-4695-8fa8-993db3be8ad4) |

### خصوصیت DBCalculateFormula:

|  |
| --- |
| توضیحات |
| فرمول محاسباتی خصوصیت در پایگاه داده هدف است(UpdateColumnsFromTargetDB.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff) |
| فیلد محاسباتی فقط خواندنی است و بروزرسانی نمی شود (EditQueryItemManager.GetUpdateQueryItems(IsDBCalculatedColumn): be3c95de-74cf-4838-aa97-c6809a1b6f29). |
| تعیین اینکه ستون یک خصوصیت ساده است یا خیر (ColumnDTO.IsSimpleColumn: edeace71-ffce-4695-8fa8-993db3be8ad4)  چالش : تاثیرش در فرم ورود اطلاعات چیست؟ |

هر ستون بر حسب نوع می تواند در مدل فرارداده به موجودیتهای دیگری نیز اختصاصی شود که این انواع ستونهای رشته ای، عددی، تاریخ، زمان و تاریخ/زمان می باشد. در ادامه به این انواع اشاره شده است.

## ستون رشته ای:

برای این نوع ستون موجودیت سیستمی StringColumnType تولید می شود. این اتفاق در بروزرسانی مدل از پایگاه داده می افتد. (UpdateColumnsFromTargetDB.UpdateColumnsFromTargetDB: 6c99db26-bb1b-40df-ba63-1e9a8bbe5eff >> BizColumn.SetColumnDataTypeFromTargetDB: fea6234f-fc47-4548-8d93-5e5127e74877 >> BizColumn.CreateNewStringColumn: c3583c1c-dcc2-42c1-979e-aa893da6b6e7)

ستونهای رشته ای دارای خصوصیات زیر می باشند.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| MaxLength | طول ماکزیمم رشته؛ |
| MinLength | حداقل طول رشته برای اعتبارسنجی؛ |
| Format | فرمت مورد قبول داده برای اعتبارسنجی؛ |

### خصوصیت MaxLength :

|  |
| --- |
| توضیحات |
| این خصوصیت مستقیماً از پایگاه داده هدف بروزرسانی می شود (BizColumn.SyncColumnTypesDBProperties: 608254b4-72f0-419f-a038-ad4ff657a6ff)؛ نمونه : خصوصیت کد (Code) در موجودیت دفتر (Office) |
| بر حسب طول رشته، در تولید ظاهر ورود اطلاعات خصوصیت یا همان ColumnUISetting ، انواع زیر در نظر گرفته می شوند (BizEntityUIComposition.AddColumnUICompositionItem: ad8db1ea-07d5-481a-95fd-5c93e6562e87) :   * رشته کوتاه : رشته هایی برای تعریف مقادیر مثل نام و نام خانوادگی. به صورت پیش فرض طول این ستونها کمتر یا مساوی 64 کاراکتر فرض می شود و ColumnUISetting با یک ستون معمولی و یک ردیف تولید می شود (ColumnUISetting.UIColumnsType = Enum\_UIColumnsType.Normal) . نمونه : نام در موجودیت شخص حقیقی (RealPerson) * رشته متوسط : رشته هایی برای تعریف مقادیر مثل عنوان یک پرونده. به صورت پیش فرض طول این ستونها بیشتر از 64 و کمتر یا مساوی 256 کاراکتر فرض می شود و ColumnUISetting با یک ستون نیم صفحه و یک ردیف تولید می شود (ColumnUISetting.UIColumnsType = Enum\_UIColumnsType.Half). نمونه : نام در موجودیت شخص حقوقی (LegalPerson) * رشته بلند : رشته هایی برای تعریف مقادیر مثل آدرس. به صورت پیش فرض طول این ستونها بیشتر از 256 فرض می شود و ColumnUISetting با یک ستون تمام صفحه ایجاد می گردد، اگر طول رشته کوچکتر یا مساوی 1024 بود ColumnUISetting یک ردیفی، اگر بزرگتر از 1024 بود (و نامحدود نبود) دو ردیفی و اگر طول رشته نامحدود بود ColumnUISetting سه ردیفی می باشد (ColumnUISetting.UIColumnsType = Enum\_UIColumnsType.Full). نمونه : آدرس در موجودیت GenericPersonAddress و یا توضیحات در موجودیت درخواست سرویس (ServiceRequest) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت MinLength :

|  |
| --- |
| توضیحات |
| این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است(BizColumnUpdateStringColumnType: bf58c975-a36d-4a12-b7a3-1cbf252eff52)؛ نمونه : خصوصیت کد (Code) در موجودیت دفتر (Office) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت Format :

|  |
| --- |
| توضیحات |
| این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است(BizColumnUpdateStringColumnType: bf58c975-a36d-4a12-b7a3-1cbf252eff52)؛ نمونه : خصوصیت آدرس ایمیل در موجودیت شخص (GenericPerson) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

## ستون عددی :

برای این نوع ستون موجودیت سیستمی NumericColumnType در زمان بروزرسانی مدل از پایگاه داده هدف تولید می شود (BizColumn.CreateNewNumericColumn: e64b16d0-bfc0-46fd-9116-67b59e86132f)

ستونهای عددی دارای خصوصیات زیر می باشند.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Precision | طول بخش صحیح عدد |
| Scale | طول بخش اعشاری عدد |
| MinValue | حداقل مقدار برای اعتبارسنجی؛ این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است(c99a93a8-756a-42a2-a9f2-d346c3f0299b). |
| MaxValue | حداکثر مقدار برای اعتبارسنجی؛ این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است(c99a93a8-756a-42a2-a9f2-d346c3f0299b). |
| Delimiter | جداکننده سه رقمی عدد؛ |

### خصوصیت Precision:

|  |
| --- |
| توضیحات |
| در بروزرسانی جداول پایگاه داده هدف شناسایی شده و مقدار آن جایگزین می شود(BizColumn.SyncColumnTypesDBProperties: 608254b4-72f0-419f-a038-ad4ff657a6ff). |
| از این خصوصیت در تولید کنترلهای ورود اطلاعات عددی استفاده می شود (NumericTextBoxHelper.NumericTextBoxHelper: f8981e30-c94d-44be-80e0-0ba321de6d67) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت Scale:

|  |
| --- |
| توضیحات |
| در بروزرسانی جداول پایگاه داده هدف شناسایی شده و مقدار آن جایگزین می شود(BizColumn.SyncColumnTypesDBProperties: 608254b4-72f0-419f-a038-ad4ff657a6ff). |
| از این خصوصیت در تولید کنترلهای ورود اطلاعات عددی استفاده می شود (NumericTextBoxHelper.NumericTextBoxHelper: f8981e30-c94d-44be-80e0-0ba321de6d67) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت MinValue:

|  |
| --- |
| توضیحات |
| این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است (BizColumn.UpdateNumericColumnType: c99a93a8-756a-42a2-a9f2-d346c3f0299b) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت MaxValue:

|  |
| --- |
| توضیحات |
| این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است (BizColumn.UpdateNumericColumnType: c99a93a8-756a-42a2-a9f2-d346c3f0299b) |
| از این خصوصیت در اعتبارسنجی فرمهای ورود اطلاعات استفاده می شود (UIValidationManager.ValidateSimpleColumn: 54ed40be-5fa9-46ee-a27b-2d60cd226e9e) |

### خصوصیت Delimiter:

|  |
| --- |
| توضیحات |
| این خصوصیت در برنامه مدیریت فراداده قابل مقداردهی است (BizColumn.UpdateNumericColumnType: c99a93a8-756a-42a2-a9f2-d346c3f0299b) |
| از این خصوصیت در تولید کنترلهای ورود اطلاعات عددی استفاده می شود (NumericTextBoxHelper.NumericTextBoxHelper: f8981e30-c94d-44be-80e0-0ba321de6d67) |

بطور کلی در تولید ظاهر خصوصیات عددی به صورت کلی، به صورت پیش فرض ColumnUISetting با یک ستون معمولی و یک ردیف ایجاد می گردد. ( BizEntityUIComposition.AddColumnUICompositionItem: ad8db1ea-07d5-481a-95fd-5c93e6562e87)

نمونه هایی از خصوصیات عددی در زیر اشاره شده است:

عدد بدون اعشار : ستون شماره شناسنامه (BirthCertificateNumber) در موجودیت شخص حقیقی (RealPerson)

عدد اعشاری: ستون نمره (UserRate) در موجودیت خلاصه سرویس (ServiceConclusion)

عدد مبلغ: ستون مبلغ (Price) در موجودیت خدمات اضافه (ServiceAdditionalItem)

## ستون تاریخ/زمان :

ستونهای تاریخ و زمان در مدل فراداده به سه دسته تقسیم می شوند. ستونهای فقط تاریخ، ستونهای فقط زمان و ستونهای تاریخ و زمان با هم. البته عموماً نوع داده این سه ستون با یکدیگر متفاوت است امّا در سیستم های فارسی از ستون رشته برای نگهداری تاریخ و زمانهای شمسی استفاده می گردد. در ادامه به بررسی هر کدام از این انواع ستون تاریخ و زمان بصورت تخصصی پرداخته می شود.

## ستون تاریخ :

برای این نوع ستون موجودیت سیستمی DateColumnType تولید می شود. همانطور که پیشتر ذکر شد این نوع ستون می تواند معادلی از نوع تاریخ میلادی در پایگاه داده داشته باشد و یا معادلی از نوع رشته. ستونهای با نوع تاریخ میلادی از پایگاه داده هدف شناسایی و ایجاد می شوند (BizColumn.CreateNewDateTimeColumn: 5a4645cf-0487-48d5-89bb-65590717ebfd). برای شناسایی تاریخ های رشته ای تاریخ در پایگاه داده هدف نیز بررسی هایی به هنگام بروزرسانی انجام می شود(BizColumn.SetColumnDataTypeFromTargetDB: fea6234f-fc47-4548-8d93-5e5127e74877). لازم به ذکر است که برای این نوع ستونها به جهت رشته و تاریخ بودن همزمان هر دو نوع StringColumnType و DateColumnType در مدل تولید میگردد(BizColumn.CreateNewStringDateTimeColumn: e740a7d8-1456-422f-b766-2b8bb6df4790). البته برای ستونهای رشته/تاریخی که بصورت خودکار شناسایی نمی شوند در برنامه مدیریت فراداده قابلیت تبدیل انواع رشته به تاریخ (BizColumn.ConvertStringColumnToDateTimeColumn: 559a48c0-417b-4813-8b9e-b6ad00bdd936) و بالعکس (BizColumn.ConvertColumnToStringColumnType: 8351e66f-a105-44ba-8a5c-7715aa287708) وجود دارد. بصورت کلی موجودیت DateColumnType دارای خصوصیات زیر است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| ShowMiladiDateInUI | اینکه در واسط کاربری کنترل تاریخ شمسی نمایش داده شود یا میلادی (DateTimePickerHelper: cbeb78f5-f3ac-41d8-b615-c2f50657509c) ( فارق از اینکه نوع ستون در پایگاه داده تاریخ است یا رشته شمسی یا میلادی) |
| DBValueIsString | اینکه ستون در پایگاه داده از نوع تاریخ است و یا رشته؟ |
| DBStringValueIsMiladi | آیا مقدار که رشته است میلادی است یا شمسی؟ |

خصوصیت ShowMiladiDateInUI :

|  |
| --- |
| توضیحات |
| مقدار این خصوصیت در درجه اول مستقیماً به روی ستون در برنامه مدیریت فرارداده قابل تعیین شدن است (BizColumn.UpdateDateColumnType: 1589bf82d5ee) |
| در صورت عدم مقداردهی مستقیم روی ستون (Null بودن مقدار)، از تنظیمات پایگاه داده در برنامه مدیریت فراداده خوانده می شود. در آنجا هم مقدار اولیه False است بنابراین بصورت پیش فرض برای تمامی ستونهای تاریخ در واسط کاربری تاریخ شمسی نمایش داده می شود (BizColumn.ToDateColumTypeDTO: 5bd4e7ec-d66f-4476-a4e7-0c4c410b98b1). |

### خصوصیت DBValueIsString :

|  |
| --- |
| توضیحات |
| اینکه مقدار خصوصیت در پایگاه داده هدف رشته است یا خیر از خصوصیت OriginalTypeEnum ستونها نیز قابل برداشت است امّا اینجا نیز به صورت اختصاصی ذخیره می شود تا رشته بودن و نبودن ستون تاریخ در پایگاه داده راحتتر قابل تشخیص باشد(BizColumn.CreateNewDateTimeColumnOriginallyString: e0a2dd6f-ec52-4430-b15f-9e272e60f7cb) |
| کاربردش در کدهای واسط کاربری است (DateTimePickerHelper: cbeb78f5-f3ac-41d8-b615-c2f50657509c) |

### خصوصیت DBStringValueIsMiladi :

|  |
| --- |
| توضیحات |
| این خصوصیت در صورت True بودن خصوصیت DBValueIsString معنی پیدا میکند. به هنگام ایجاد ستون رشته/تاریخ در صورت وجود داده در پایگاه داده هدف توسط قطعه کدی به صورت پیش فرض تشخیص داده می شود(BizColumn.CreateNewDateTimeColumnOriginallyString: e0a2dd6f-ec52-4430-b15f-9e272e60f7cb). |
| البته این خصوصیت در برنامه مدیریت فراداده نیز قابل مقداردهی است (BizColumn.UpdateDateColumnType: 1589bf82d5ee) |
| به هنگام استفاده از این خصوصیت در واسط کاربری، در صورت مشخص نبودن مقدار (Null بودن مقدار)، از تنظیمات پایگاه داده در برنامه مدیریت فراداده خوانده می شود. در آنجا هم مقدار اولیه False است بنابراین بصورت پیش فرض برای تمامی ستونهای تاریخ رشته مقادیر به صورت شمسی فرض می شود (BizColumn.ToDateColumTypeDTO: 5bd4e7ec-d66f-4476-a4e7-0c4c410b98b1). |
| کاربردش در کدهای واسط کاربری است (DateTimePickerHelper: cbeb78f5-f3ac-41d8-b615-c2f50657509c) |
| دقت شود که از این خصوصیت برای ذخیره سازی و یا دریافت مقادیر رشته ای تاریخ در کد ها استفاده می شود و نوع نمایش تاریخ میلادی یا شمسی در واسط کاربری وابسته به همان خصوصیت ShowMiladiDateInUI می باشد. |

مثالهایی از انواع ستون تاریخ و تنظیمات مربوطه:

* ستون از نوع تاریخ میلادی در پایگاه داده و نمایش میلادی (ShowMiladiDateInUI برابر true) : ستون تاریخ بروزرسانی(UpdateDate) در جدول مورد سرویس (ServiceItem)
* ستون از نوع تاریخ میلادی در پایگاه داده و نمایش شمسی (ShowMiladiDateInUI برابر true نباشد) : خصوصیت تاریخ تولد(BirthDate) در موجودیت شخص حقیقی (RealPerson)
* ستون از نوع رشته در پایگاه داده با مقادیر شمسی و نمایش میلادی(DBValueIsStringMiladi برابر true نباشد و ShowMiladiDateInUI برابر true باشد) : خصوصیت تاریخ (PersianDate) در موجودیت درخواست سرویس (ServiceRequest)
* ستون از نوع رشته در پایگاه داده با مقادیر میلادی و نمایش شمسی (DBValueIsStringMiladi برابر true) : خصوصیت تاریخ خودکار (AutoDate) در موجودیت بررسی درخواست سرویس (ServiceRequestReview)

## ستون زمان :

برای این نوع ستون موجودیت سیستمی TimeColumnType تولید می شود. همانطور که پیشتر ذکر شد این نوع ستون می تواند معادلی از نوع زمان در پایگاه داده داشته باشد و یا معادلی از نوع رشته. ستونهای با نوع زمان از پایگاه داده هدف شناسایی و ایجاد می شوند (BizColumn.CreateNewDateTimeColumn: 65590717ebfd). برای شناسایی تاریخ های رشته ای تاریخ در پایگاه داده هدف نیز بررسی هایی به هنگام بروزرسانی انجام می شود(BizColumn.SetColumnDataTypeFromTargetDB: 5e5127e74877). لازم به ذکر است که برای این نوع ستونها به جهت رشته و زمان بودن همزمان هر دو نوع StringColumnType و TimeColumnType در مدل تولید میگردد(BizColumn.CreateNewStringDateTimeColumn: 2b8bb6df4790). البته برای ستونهای رشته/زمان که بصورت خودکار شناسایی نمی شوند در برنامه مدیریت فراداده قابلیت تبدیل انواع رشته به تاریخ (BizColumn.ConvertStringColumnToDateTimeColumn: b6ad00bdd936) و بالعکس (BizColumn.ConvertColumnToStringColumnType: 7715aa287708) وجود دارد. بصورت کلی موجودیت TimeColumnType دارای خصوصیات زیر است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| DBValueIsString | اینکه ستون در پایگاه داده از نوع زمان است و یا رشته؟ |
| DBStringValueTimeFormat | فرمت مقدار زمان در صورتی که رشته است؛ این خصوصیت در صورت True بودن خصوصیت DBValueIsString معنی پیدا میکند. |

### خصوصیت DBValueIsString :

|  |
| --- |
| توضیحات |
| اینکه مقدار خصوصیت در پایگاه داده هدف رشته است یا خیر از خصوصیت OriginalTypeEnum ستونها نیز قابل برداشت است امّا اینجا نیز به صورت اختصاصی ذخیره می شود تا رشته بودن و نبودن ستون زمان در پایگاه داده راحتتر قابل تشخیص باشد(BizColumn.CreateNewDateTimeColumnOriginallyString: 9e272e60f7cb) |
| کاربردش در کدهای واسط کاربری است (DateTimePickerHelper: c2f50657509c) |

### خصوصیت DBStringValueIsMiladi :

|  |
| --- |
| توضیحات |
| این خصوصیت بصورت Enum است و مقادیر ممکن در آن، Unknown (نامشخص)، Hours24(24ساعته)، AMPMMiladi(12 ساعته میلادی) و AMPMShamsi (12 ساعته شمسی) می باشد. به هنگام ایجاد ستون رشته/زمان در صورت وجود داده در پایگاه داده هدف توسط قطعه کدی به صورت پیش فرض تشخیص داده می شود(BizColumn.CreateNewDateTimeColumnOriginallyString: 9e272e60f7cb). |
| البته این خصوصیت در برنامه مدیریت فراداده نیز قابل مقداردهی است (BizColumn.UpdateTimeColumnType: 247a404e04f7) |
| به هنگام استفاده از این خصوصیت در واسط کاربری، در صورت مشخص نبودن مقدار (Null بودن مقدار)، از تنظیمات پایگاه داده در برنامه مدیریت فراداده خوانده می شود. در صورتی که آنجا نیز تعیین نشده باشد بصورت پیش فرض مقادیر به صورت "24ساعته" فرض می شود (BizColumn.ToTimeColumTypeDTO: 8f3baefaf4fd). |
| دقت شود که از این خصوصیت برای ذخیره سازی و یا دریافت مقادیر رشته ای زمان در کد ها استفاده می شود و نمایش زمان در واسط کاربری همواره به صورت 24 ساعته انجام می شود. |

مثالهایی از انواع ستون زمان و تنظیمات مربوطه :

* ستون از نوع زمان در پایگاه داده : ستون زمان بروزرسانی (UpdateTime) در موجودیت مورد سرویس (ServiceItem)
* ستون از نوع رشته در پایگاه داده و مقادیر 12 ساعته شمسی (DBValueStringTimeFormat برابر با AMPMShamsi) : فیلد زمان (Time) در موجودیت درخواست سرویس (ServiceRequest)
* ستون از نوع رشته در پایگاه داده و مقادیر 24 ساعته (DBValueStringTimeFormat برابر با Hours24) : خصوصیت زمان خودکار (AutoTime) در موجودیت بررسی درخواست سرویس (ServiceRequestReview)

## ستون تاریخ و زمان :

برای این نوع ستون موجودیت سیستمی DateTimeColumnType تولید می شود. همانطور که پیشتر ذکر شد این نوع ستون می تواند معادلی از نوع تاریخ و زمان در پایگاه داده داشته باشد و یا معادلی از نوع رشته. ستونهای با نوع تاریخ و زمان از پایگاه داده هدف شناسایی و ایجاد می شوند (BizColumn.CreateNewDateTimeColumn: 65590717ebfd). برای شناسایی تاریخ های رشته ای تاریخ در پایگاه داده هدف نیز بررسی هایی به هنگام بروزرسانی انجام می شود(BizColumn.SetColumnDataTypeFromTargetDB: 5e5127e74877). لازم به ذکر است که برای این نوع ستونها به جهت رشته و تاریخ و زمان بودن همزمان هر دو نوع StringColumnType و DateTimeColumnType در مدل تولید میگردد(BizColumn.CreateNewStringDateTimeColumn: 2b8bb6df4790). البته برای ستونهای رشته/تاریخ و زمان که بصورت خودکار شناسایی نمی شوند در برنامه مدیریت فراداده قابلیت تبدیل انواع رشته به تاریخ (BizColumn.ConvertStringColumnToDateTimeColumn: b6ad00bdd936) و بالعکس (BizColumn.ConvertColumnToStringColumnType: 7715aa287708) وجود دارد. بصورت کلی موجودیت DateTimeColumnType دارای خصوصیات زیر است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| ShowMiladiDateInUI | اینکه در واسط کاربری کنترل تاریخ شمسی نمایش داده شود یا میلادی (DateTimePickerHelper: cbeb78f5-f3ac-41d8-b615-c2f50657509c) ( فارق از اینکه نوع ستون در پایگاه داده تاریخ است یا رشته شمسی یا میلادی) |
| DBValueIsString | اینکه ستون در پایگاه داده از نوع تاریخ است و یا رشته؟ |
| DBStringValueIsMiladi | آیا مقدار که رشته است میلادی است یا شمسی؟ |
| DBStringValueTimeFormat | فرمت مقدار زمان در صورتی که رشته است؛ این خصوصیت در صورت True بودن خصوصیت DBValueIsString معنی پیدا میکند. |

همانگونه که مشخص است خصوصیات ستونهای تاریخ و زمان ترکیبی از خصوصیات نوع تاریخ و نوع زمان است. بنابراین برای توضیحات تکمیلی این خصوصیات به همان توضیحات بخشهای قبلی مراجعه شود.

مثالهایی از انواع ستون تاریخ/زمان و تنظیمات مربوطه :

* ستون از نوع تاریخ/زمان در پایگاه داده و نمایش میلادی (ShowMiladiDateInUI برابر true) : فیلد تاریخ/زمان (DateTime) در موجودیت درخواست قطعه (RequestProductPart)
* ستون تاریخ/زمان از نوع رشته ای با مقادیر تاریخ میلادی و زمان 12 ساعته شمسی و نمایش تاریخ شمسی (DBValueIsStringMiladi برابر true و DBValueStringTimeFormat برابر با AMPMShamsi) : فیلد تاریخ/زمان رشته (StringDateTime) در موجودیت درخواست قطعه (RequestProductPart)
* ستون تاریخ/زمان از نوع رشته ای با مقادیر تاریخ شمسی و مقدار 12 ساعته میلادی و نمایش میلادی (DBValueIsStringMiladi برابر true نباشد و DBValueStringTimeFormat برابر با AMPMMiladi باشد و ShowMiladiDateInUI برابر true باشد) : فیلد تاریخ و زمان بروزرسانی (UpdateDateTime) در موجودیت خلاصه سرویس (ServiceConclusion)

## ستون Boolean:

برای این نوع ستون موجودیت سیستمی مخصوصی تولید نمی شود. تنها نکته اینست که اگر ستون Null پذیر باشد در واسط کاربری چک باکس سه حالتی (با مقادیر true، false و null) و در غیر اینصورت چک باکس دو حالتی(با مقادیر true، false) تولید می شود (CheckBoxHelper: c5150319d9e8).

مثال چک باکس سه حالتی : فیلد تایید شده (Confirmed) در جدول آیتم خلاصه سرویس (ServiceConclusionItem)

مثال چک باکس دو حالتی : فیلد تسویه شده (IsCleared) در جدول خلاصه سرویس (ServiceConclusion)

## ستون لیست مقادیر :

برخی ستونها فارق از اینکه از نوع رشته و یا عدد می باشند نیاز است مقادیرشان از بازه ای از مقادیر قابل انتخاب در واسط کاربری تعریف شوند. لیست این مقادیر برای ستونها در فرمی مخصوص در برنامه مدیریت فراداده انجام می شود (BizColumnValueRange.UpdateColumnValueRange: 1abc824a1aa2). اساساً در برای این نیازمندی در مدل موجودیتهای اختصاصی ColumnValueRange و ColumnValueRangeDetails استفاده می شوند. موجودیت ColumnValueRange که به صورت یک به یک با ستونهای تعریف شده در ارتباط است و ColumnValueRangeDetails نیز بصورت چند به یک با ColumnValueRange مرتبط است. لیست مقادیر در موجودیت ColumnValueRangeDetails نگهداری می شوند.

### ساختار ColumnValueRangeDetails :

لیست مقادیر که ارتباطی چند به یک با یک ستون دارد (از طریق موجودیت واسط ColumnValueRange) دارای خصوصیات زیر است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| KeyTitle | عنوان مقدار جهت نمایش در واسط کاربری. این خصوصیت اختیاری است و در صورت عدم مقداردهی همان مقادیر موجود در Value در واسط کاربری نمایش داده می شوند. |
| Value | خصوصیت اصلی است. مقداری که در پایگاه داده هدف ذخیره و بازیابی می شود. |
| Tag | کلیدی برای دسته بندی مقادیر در صورت نیاز به فیلتر کردن لیست مقادیر. |

در برنامه مدیریت فراداده می توان لیست مقادیر را در صورت وجود به صورت پیش فرض نیز از پایگاه داده هدف استخراج کرد(frmColumnValueRange.btnImportKeyValues\_Click: 1489ef28f581).

### ساختار ColumnValueRange :

هر ColumnValueRange ارتباطی یک به یک با ستونها دارد. تنها خصوصیاتی که در این موجودیت نگهداری می شوند اطلاعات فیلد مرجع به منظور فیلتر کردن مقادیر می باشد. همانطور که اشاره شد، توسط خصوصیت Tag می توان لیست مقادیر را در صورت نیاز با توجه به مقدار خصوصیت دیگری فیلتر نمود. در واقع در لیست مقادیر قابل انتخاب برای ستون، تنها مواردی نمایش داده می شوند که خصوصیت Tag آنها برابر با مقدار فیلد مرجع تعیین شده باشد. برای مثال می توان به لیست مقادیر خصوصیت نقش تخصصی (EmployeeRole2) در موجودیت کارمند (Employee) اشاره کرد که با انتخاب فیلد مرجع نقش عمومی (EmployeeRole) فیلتر می شوند. مسیر و عنوان ستون فیلد مرجع در موجودیت ColumnValueRange نگهداری می شود که در زیر آمده است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| EntityRelationshipTailID | رشته رابطه ای ستون مرجع که لیست مقادیر با توجه به مقدار آن ستون فیلتر شوند. |
| TagColumnID | ستون مرجع که لیست مقادیر با توجه به مقدار آن ستون فیلتر شوند. |

### لیست مقادیر در واسط کاربری :

|  |
| --- |
| توضیحات |
| برای ستونهای دارای ColumnValueRange، کنترلی کمبو مانند ایجاد می گردد (KeyValueControlHelper: aca04d5175f8) |
| در صورت نیاز به فیلتر لیست بر اساس فیلد مرجع از وضعیتهای داینامیک فرم ورود اطلاعات کمک گرفته می شود (UIActionActivityManager.CheckColumnValueRangeStates: 403d341eba76) |
| در اعتبارسنجی فیلدهای فرمهای ورود اطلاعات لیست مقادیر یک ستون کنترل می شود(UIValidationManager.ValidateSimpleColumn: 2d60cd226e9e) |

**چالش** : در صورتی که خصوصیتی در واسط کاربری دارای مقدار باشد و مقدار در لیست مقادیر(چه فیلتر شده و چه فیلتر نشده) موجود نباشد چه اتفاقی برای مقدار خصوصیت می افتد؟

# روابط در مدل فراداده

بخش مهم دیگری از مدل فرداداده تعریف روابط بین موجودیتها در مدل می باشد. این روابط عمدتاً عیناً از پایگاه داده هدف استخراج می شوند، امّا امکان تعریف روابط بین موجودیتها نیز در مدل وجود دارد. نکته مهم درباره روابط، ارتقاء آنها فراتر از لایه فیزیکی پایگاه داده و تعریف خصوصیات و مفاهیم سطح بالاتر برای هر رابطه است. در ادامه به این مفاهیم درباره روابط پرداخته می شود.

اولین گام در تحلیل روابط شناسایی نوع رابطه بر حسب تعدد رابطه (یک به یک، یک به چند) می باشد. سپس روابط سطح بالاتر به مانند روابط ارث بری و اتحاد نیاز به شناسایی و دسته بندی دارند.

در اولین گام، تمامی روابط به صورت خودکار از پایگاه داده هدف استخراج می شوند و سپس بعضی از انواع رابطه در صورت امکان از نوع ستونهای درگیر و یا داده های موجود به صورت پیش فرض شناسایی می شوند. البته چه در فاز استخراج و چه در برنامه مدیریت فراداده در صورت نیاز امکان تغییر انواع رابطه وجود دارد.

به صورت کلی هر رابطه به در مدل فراداده به صورت دو طرفه یا جفت تعریف می شود. دلیل آن سهولت در اختصاصی نمودن تعاریف خصوصیات هر سر رابطه در ورود اطلاعات موجودیت ها می باشد. انواع رابطه قابل تعریف در مدل فراداده رابطه یک به یک، رابطه یک به چند (وعکس آن چند به یک)، رابطه چند به چند، رابطه ارث بری و رابطه اتحاد می باشد. چهار نکته در مورد ورود اطلاعات روابط دارای اهمیت می باشد.

* منظور از ورود اطلاعات رابطه، مقداردهی فیلد های کلید خارجی رابطه بر اساس کلیدهای اصلی موجودیت اصلی و ایجاد ارتباط بین دو موجودیت می باشد.
* هر رابطه را میتوان از سمت هر موجودیت طرف رابطه ورود اطلاعات کرد امّا معمولاً در هر برنامه بنابر منطق حاکم (مانند تقدم و تأخر زمانی داده ها، استقلال داده، تعدد روابط موجودیت و ...)، ورود اطلاعات رابطه تنها از موجودیت یک طرف رابطه امکانپذیر می باشد. برای مثال روابط یک به چندی که طرف یک آن داده مرجعی مانند شهر می باشد معمولاً از طرف موجودیت چند و با انتخاب شهر مربوطه انجام می شود. در مقابل در فرم مخصوص موجودیت شهر، تنها به تعریف شهرها پرداخته می شود و به روابط آن با سایر موجودیت ها اعتنایی نمی شود. در نقطه مقابل رابطه یک فرد با آدرس های مربوطه به فرد فرض شود. با وجود اینکه اینبار نیز رابطه یک به چند امّا موجودیت آدرس (که طرف چند است) هویتی مستقل ندارد و تنها به همراه طرف یک ورود اطلاعات می شود.
* اینکه موجودیت هر طرف رابطه از قبل موجود بوده و یا در لحظه به همراه رابطه تعریف می شود نیاز به بررسی دارد. داده فرم جاری خود می تواند داده ای جدید باشد یا داده از قبل موجود. بر قراری ارتباط با موجودیت طرف دیگر نیز سه حالت دارد، می توان موجودیت طرف دیگر رابطه را تنها انتخاب نمود، می توان تنها ورود اطلاعات نمود و یا هم اجازه انتخاب داده و هم ورود اطلاعات.
* در صورت امکان ورود اطلاعات طرف دیگر رابطه، خود این ورود اطلاعات می تواند به صورت مستقیم یعنی با نمایش فرم موجودیت طرف دوم در فرم موجودیت اول باشد و یا به صورت میانبری به فرم موجودیت طرف دوم باشد (طرف دوم در فرم جداگانه ای باز شود).

برخی از خصوصیات تقریباً برای همه انواع رابطه یکسان می باشند که در زیر به آنها اشاره شده است:

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Name | نام رابطه |
| Alias | عنوان رابطه |
| TypeEnum | نوع رابطه |
| IsOrginal | رابطه استخراج شده از پایگاه داده است؟ |
| SearchInitially | آیا داده های طرف دیگر بصورت خودکار لیست شوند. |
| IsOtherSideMandatory | آیا طرف دیگر رابطه اجباری است؟ |
| IsOtherSideCreatable | آیا طرف دیگر رابطه قابل ورود اطلاعات است؟ |
| IsOtherSideDirectlyCreatable | آیا طرف دیگر که ورود اطلاعات می شود بصورت مستقیم است یا میانبر؟ |
| IsNotSkippable | آیا رابطه به هنگام ورود اطلاعات به تشخیص سامانه قابل صرف نظر است؟ |
| Created | آیا رابطه بصورت دستی ایجاد شده است؟ |
| IsDisabled | آیا رابطه غیر فعال شده است؟ |
| IsReadonly | آیا رابطه فقط خواندنی است؟ |
| DataEntryEnabled | آیا رابطه در فرم ورود اطلاعات نمایش داده شود. |
| DBDeleteRule | قانون بروزرسانی رابطه در پایگاه داده |
| DBUpdateRule | قانون حذف رابطه در پایگاه داده |
| DeleteOption | نوع شرط حذف در پایگاه داده |

## کاربرد خصوصیات رابطه در فرم های ورود اطلاعات :

### خصوصیت IsOtherSideMandatory :

|  |
| --- |
| توضیحات |
| * اجباری بودن روابط در اولین گام از پایگاه داده هدف استخراج می شود (BizRelationship.UpdateRelationshipInModel: 93500ebd3e5e) |
| * افزودن ستاره به عناوین رابطه (BaseEditEntityArea.GenerateControls: ddfdfcbe4f89) * تولید تولید رنگ و تولتیپ برای رابطه (ChildRelationshipInfo.SetControlUIDetails: a579903e12bd) |
| * تعیین قابل صرف نظر بودن رابطه (BizTableDrivedEntity.ShouldSkipRelationshipDepended: 57991005621a) |
| * کنترل روابط اجباری امّا غایب (UIValidationManager.CheckAccessValidation: 3b7eb162083e) |
| * کنترل اجباری بودن ورود اطلاعات رابطه (UIValidationManager.ValidateRelationshipColumn: dec468052171) |

### خصوصیت IsOtherSideCreatable :

|  |
| --- |
| توضیحات |
| * تعیین نوع IntracionMode رابطه برای واسط کاربری (BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7) |

### خصوصیت IsOtherSideDirectlyCreatable :

|  |
| --- |
| توضیحات |
| * به صورت کلی در کلاسهای تولید پیش فرض ترکیب واسط کاربری (BizEntityUIComposition.GetOrCreateEntityUIComposition: fdb08b6bb258) |
| * تعیین نوع IntracionMode رابطه برای واسط کاربری (BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7) |

### خصوصیت IsNotSkippable :

|  |
| --- |
| توضیحات |
| * این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است. (BizRelationship.ToRelationshipDB: c2cecbba5f18) * تعیین قابل صرف نظر بودن رابطه برای واسط کاربری (BizTableDrivedEntity.ShouldSkipRelationshipDepended: 57991005621a) |

### خصوصیت Removed :

|  |
| --- |
| توضیحات |
| * تعیین حذف شده بودن رابطه از مدل. این خصوصیت تنها از پایگاه داده هدف و در صورت حذف شدن رابطه از پایگاه داده هدف قابل مقداردهی می باشد (BizRelationship.UpdateRelationshipInModel: 58069fb34622). |
| * دریافت لیست رابطه ها بصورت کلی (BizRelationship.GetAllRelationships: 8a8242e6eb63). |
| * دریافت لیست روابط یک موجودیت (BizRelationship.GetEnabledRelationships: 84d63337f284). |
| * تعیین کلید خارجی بودن یک ستون (BizColumn.ToColumnDTO: 3fad23169aeb). |

### خصوصیت IsDisabled :

|  |
| --- |
| توضیحات |
| * این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است. نکته اینست که این خصوصیت مستقیم به روی رابطه نگهداری نمی شود بلکه مستقیماً به روی خصوصیت IsDisabled ستون کلید خارجی رابطه ذخیره و خوانده می شود. (BizRelationship.ToRelationshipDB: c2cecbba5f18 و BizRelationship.ToRelationshipDTO: f30042893bc1) |
| * در خواندن رابطه های مدل فراداده نیز بسته به اینکه رابطه های فعال مورد نیازند یا تمامی رابطه ها از این خصوصیت برای فیلتر کردن استفاده می شود (BizRelationship.GetAllEnabledRelationships: ba141918b2bb). |
| * دریافت لیست روابط یک موجودیت (BizRelationship.GetEnabledRelationships: 84d63337f284). |

### خصوصیت DataEntryEnabled :

|  |
| --- |
| توضیحات |
| * تعیین نمایش رابطه در فرم ورود اطلاعات (BizTableDrivedEntity.CheckDataEntryPermission: 8be394c313b9). |
| * این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است. نکته اینست که این خصوصیت برای دو طرف رابطه می تواند متفاوت باشد. برای سمت اصلی به فرعی رابطه در مدل فراداده به روی PKToFKDataEntryEnabled ذخیره و بازیابی می شود و برای طرف فرعی به اصلی، خصوصیت مستقیم به روی رابطه نگهداری نمی شود بلکه مستقیماً به روی خصوصیت DataEntryEnabled ستون کلید خارجی رابطه ذخیره و خوانده می شود(BizRelationship.ToRelationshipDB: c2cecbba5f18). |
| * استفاده در کلاسهای تولید پیش فرض ترکیب واسط کاربری (BizEntityUIComposition.GetOrCreateEntityUIComposition: 33f45647385a) |
| * امکان تعریف فیلتر رابطه تنها به روی روابط قابل ورود اطلاعات امکانپذیر است (frmEntityRelationships.dtgRelationships\_ContextMenuOpening: 1ce825701244) |
| * کنترل روابط اجباری امّا غایب (UIValidationManager.CheckAccessValidation: 3b7eb162083e) |

### خصوصیت IsReadonly :

|  |
| --- |
| توضیحات |
| * تعیین فقط خواندنی بودن رابطه. این خصوصیت توسط کاربر در برنامه مدیریت فراداده قابل مقداردهی است. نکته اینست که این خصوصیت مستقیم به روی رابطه نگهداری نمی شود بلکه مستقیماً به روی خصوصیت IsReadonly ستون کلید خارجی رابطه ذخیره و خوانده می شود. (BizRelationship.ToRelationshipDB: c2cecbba5f18 و BizRelationship.ToRelationshipDTO: f30042893bc1) |
| * فقط خواندنی بودن روابط در دریافت موجودیت ورود اطلاعات کنترل می شوند (BizTableDrivedEntity.CheckDataEntryPermission: 8be394c313b9). |
| * در فرمهای ورود اطلاعات، تعیین فقط خواندنی بودن رابطۀ داده (ChildRelationshipInfo.IsReadonly: e276ef44e0fb) |
| * افزودن تولتیپ به ظاهر رابطه (ChildRelationshipInfo.SetControlUIDetails: a579903e12bd) |

### خصوصیت Created :

|  |
| --- |
| توضیحات |
| * از این خصوصیت برای تعیین رابطه هایی که توسط کاربر در برنامه مدیریت فراداده ایجاد می شوند و نه آنهایی که از پایگاه داده هدف استخراج می شوند (BizRelationshipCreateUpdateRelationship: ab903c4acf08) . |
| * تنها رابطه هایی که توسط کاربر ایجاد شده باشند قابلیت حذف و اصلاح را در برنامه مدیریت فراداده دارند(frmConceptualRelationships.dtgRelationships\_ContextMenuOpening: 04618a5382fe و frmEntityRelationships.dtgRelationships\_ContextMenuOpening: 1ce825701244) |

# تعریف انواع رابطه

گام اول در شناسایی انواع روابط فرم استخراج روابط می باشد. در این فرم بر اساس نوع ستونهای درگیر و همچنین ارتباط داده ها (در صورت وجود) شناسایی اولیه ای بر روی روابط انجام می گردد و روابط به چند نوع یک به یک، یک به چند، ارث بری و اتحاد تقسیم می گردند (frmImportRelationships.DetermineRelationshipType: 390db5fb494d). البته در همین فرم امکان تغییر نوع روابط توسط کاربر نیز وجود دارد. پیش از ورود رابطه در مدل کنترلهایی به بوری روابط انجام می شود تا نوع رابطه تعریف شده با تعاریف فیزیکی آن رابطه در پایگاه داده همخوانی داشته باشد (frmImportRelationships.UpdateModel: a851065d8960). سپس این روابط در مدل فراداده وارد می شوند. همانطور که ذکر شد، هر رابطه فیزیکی ، به دو رابطه جهت دار تبدیل و ذخیره می شود(BizRelationship.UpdateRelationshipInModel: 93500ebd3e5e). در ادامه به اختصار انواع روابط قابل تعریف در مدل فراداده معرفی می گردند. به منظور پیچیده نشدن تعاریف انواع رابطه، کاربردهای اختصاصی هر نوع رابطه و تفاوتهای احتمالی آنها در فرم های ورود اطلاعات را در توضیحات هر بخش از فرمهای ورود اطلاعات ارائه خواهد شد.

## روابط یک به چند :

دو سناریو در ورود اطلاعات این روابط قابل فرض است. یکی ورود اطلاعات از طرف چند به طرف یک و دیگری بالعکس.

### رابطه از طرف چند به یک:

این حالت ورود اطلاعات رابطه، عمدتاً به منظور افزودن خصوصیاتی به موجودیت طرف چند در قالب موجودیت دیگر (طرف یک) می باشد. اطلاعات طرف یک معمولاً از قبل موجود بوده و انتخاب می شود. در صورت نیاز به ورود اطلاعاتِ طرف یکِ این ارتباط، فرم مخصوص طرف دیگر به صورت جداگانه فراخوانی می شود. به عبارت دیگر جزئیات طرف یک به منظور ورود اطلاعات مستقیماً در طرف چند نمایش داده نمی شوند. البته حالت مطرح شده مدل پیش فرض می باشد و تمامی تنظیمات از طریق خصوصیات ذکر شده برای هر رابطه قابل اختصاصی شدن می باشند.

مثال ورود اطلاعات طرف یک از طرف چند بصورت غیر مستقیم : ورود اطلاعات موجودیت دفتر (Office) از طریق کارمند (Employee)

مثال ورود اطلاعات طرف یک از طرف چند بصورت مستقیم : ورود اطلاعات قلم کالا (ProductItem) از طریق درخواست سرویس (ServiceRequest)

#### کاربرد های خاص:

|  |
| --- |
| توضیحات |
| * روابط چند به یک در ساخت لیست نمایش پیش فرض کاربرد دارند (BizEntityListView.GenereateDefaultListViewColumns: e0b50248b309) |
| * روابط چند به یک در ساخت لیست جستجوی پیش فرض کاربرد دارند (BizEntitySearch.GenereateDefaultSearchColumns: 9e5b4c36a21c) |

### رابطه از طرف یک به چند:

این روابط عمدتاً بر مبنای تعلق داشتن چندین مورد از موجودیت طرف چند به موجودیت طرف یک شکل می گیرد. اطلاعات طرف چند معمولاً به صورت دسته ای و وابسته به طرف یک ورود اطلاعات می شوند و بنابر ماهیت این نوع رابطه، ورود اطلاعات بصورت مستقیم همراه با موجودیت طرف یک می باشد.

مثال ورود اطلاعات طرف چند از طرف یک بصورت مستقیم: ورود اطلاعات مورد بررسی درخواست (ServiceRequestReviewItems) از طریق بررسی درخواست سرویس (ServiceRequestReview)

#### کاربرد های خاص:

|  |
| --- |
| توضیحات |
| * در ساخت موجودیت ورود اطلاعات، به روابط یک به چند بصورت صریح اشاره می شود (BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7) |

## روابط یک به یک :

این روابط معمولاً برای ارتباط دو موجودیت متفاوت که مکمل یکدیگر می باشند استفاده می شود. بنابر ماهیت پیاده سازی رابطه به هر حال یک طرف آن موجودیت اصلی که دارای کلید اصلی رابطه است قرار دارد و طرف دیگر موجودیت فرعی که از طریق کلید خارجی با موجودیت اصلی در ارتباط است. بنابراین در ورود اطلاعات این رابطه نیز به دو رابطه اصلی به فرعی و فرعی به اصلی تقسیم می گردد.

### رابطه از موجودیت اصلی به فرعی :

طرف دیگر رابطه یا همان طرف فرعی بنابر مستقل بودن یا نبودن می تواند مستقیماً ورود اطلاعات شود و یا انتخاب شود.

مثال : ورود اطلاعات دفتر تحت مسئولیت (Office) از طریق فرم کارمند (Employee)

### رابطه از موجودیت فرعی به اصلی :

موجودیت طرف اصلی معمولاً از قبل موجود می باشد، بنابراین طرف اصلی در فرم ورود اطلاعات موجودیت اصلی صرفاً انتخاب می شود.

مثال : ورود اطلاعات درخواست سرویس(ServiceRequest ) از طریق فرم بررسی درخواست سرویس (ServiceRequestReview)

مثال : ورود کارمند مسئول دفتر (Employee) از طریق دفتر(Office )

|  |
| --- |
| توضیحات |
| * روابط یک به یک فرعی به اصلی در ساخت لیست نمایش پیش فرض کاربرد دارند (BizEntityListView.GenereateDefaultListViewColumns: e0b50248b309) |
| * روابط یک به یک فرعی به اصلی در ساخت لیست جستجوی پیش فرض کاربرد دارند (BizEntitySearch.GenereateDefaultSearchColumns: 9e5b4c36a21c) |

چالش : اگر رابطه یک به یک در سمت موجودیت فرعی به روی کلید اصلی تعریف نشده باشد، حفظ جامعیت رابطه نیاز به توجه دارد زیرا ممکن است یک داده در طرف فرعی به بیش از یک داده در طرف اصلی متصل شوند. برای حل این دغدغه می توان از قوانین یکتا کننده مقادیر به روی ستونهای رابطه یک به یک در طرف فرعی استفاده نمود.

## روابط ارث بری :

روابط ارث بری، روابطی مفهومی می باشند که در سطحی بالاتر از پایگاه داده معنی می یابند. هدف از تعریف این نوع رابطه، اختصاصی نمودن خصوصیات و روابط یک موجودیت (که به آن موجودیت پدر یا مادر اطلاق می شود) در موجودیت های دیگر(موجودیتهای فرزند) و یا بالعکس اشتراک گیری از خصوصیات و روابط چند موجودیت فرزند در قالب موجودیت پدر می باشد. تفاوت عمده این رابطه با رابطه یک به یک اولاً تعریف گروه روابط به جای یک رابطه مستقل و دوماً اشتراکات ماهیتی موجودیت های درگیر در رابطه ارث بری می باشد. به عبارت دیگر عمدتاً این روابط می توانند از کلید های اصلی یکسانی استفاده کنند.

برای روابط ارث بری می توان دو خصوصیت را تعریف نمود، اول اینکه رابطه همپوشان (Overlap) است یا خیر، به این معنا که آیا موجودیت اصطلاحاً پدر (یا اصطلاحاً مادر) می تواند با بیش از یک موجودیت فرزند در ارتباط باشد و یا خیر. خصوصیت دوم شرکت عمومی در رابطه است (TotalParticipation)، به این مفهوم که آیا تمام داده های موجودیت پدر حتماً باید در رابطه ارث بری حضور یابند یا خیر. برای مثال در رابطه ارث بری بین موجودیت پدر به مانند واحد سازمانی و موجودیت های فرزند مانند اداره، شرکت، شعبه و ... خصوصیت اول تعیین می کند که آیا یک سازمانی می تواند هم شرکت و هم شعبه و هم ... باشد یا خیر. خصوصیت دوم تعیین می کند آیا هر واحد سازمانی باید در موجودیتهای فرزند اختصاصی شود و یا خیر.

در استخراج اولیه روابط، روابط ارث بری تنها در صورتی که روی کلید اصلی جداول تعریف شوند بصورت پیش فرض قابل شناسایی می باشند (frmImportRelationships.DetermineRelationshipType: 390db5fb494d). البته همانطور که ذکر شد تغییر انواع روابط در این فرم به صورت دستی نیز امکان پذیر است و می توان به صورت دستی هر چند رابطه فیزیکی را در قالب یک رابطه ارث بری دسته بندی نمود. پس از استخراج روابط نیز در برنامه مدیریت فراداده امکان تغییر مشخصات یک رابطه ارث بری وجود دارد (BizISARelationship.UpdateSuperToSubRelationships: d090c7c637f7). بطور کلی روابط مفهومی مانند ارث بری و اتحاد در فرمی مخصوص در برنامه مدیریت فراداده قابل مشاهده می باشند (frmConceptualRelationships: fdf71cf178e5)

یک رابطه ارث بری را می توان در سطح پایگاه داده به دو صورت پیاده سازی نمود :

### ارث بری به روش چند جدولی :

در این روش ارث بری، هر کدام از موجودیتهای پدر و فرزند، جدول مخصوص خود را در پایگاه داده دارا می باشند. ارتباط یک به یک بین موجودیتهای فرزند و پدر می تواند از طریق کلید اصلی موجودیت پدر و کلیدهای خارجی موجودیت فرزند تعریف شود. طبیعتاً در این روش به علت جدا بودن جداول، رابطه ارث بری شامل مجموعه ای از روابط فیزیکی موجود در پایگاه داده خواهد بود.

مثال : ارتباط بین موجودیت پدر شخص (GenericPerson) و شخص حقیقی(RealPerson) و شخص حقوقی(LegalPerson)

مثال: ارث بری بین موجودیت پدر شخص حقیقی(RealPerson) و کارمند (Employee)

مثال: ارث بری بین موجودیت پدر شخص (GenericPerson) و مشتری (Customer)

### ارث بری به روش تجمیعی در یک جدول :

در این روش تمامی موجودیت های مفهومی در یک جدول فیزیکی پیاده سازی می شوند و هر کدام مالک یک یا چند ستون مشخص می باشند. معمولاً در این روش کلید اصلی یکسان می باشد و همچنین تفکیک بین داده های انواع موجودیت ها از طریق یک یا چند فیلد "نوع" صورت می گیرد. در این روش به علت اشتراکی بودن جدول موجودیتهای ارث بری، عملاً رابطه فیزیکی در پایگاه داده تعریف نمی شود. در برنامه مدیریت فراداده می توان ضمن جداسازی موجودیتهای این نوع رابطه ارث بری، روابط بین موجودیت ها را نیز بازتعریف نمود( frmEditBaseEntity: e6c6227212d5).

مثال : ارث بری جداول گارانتی(Guarantee) به عنوان موجودیت اصلی و زیر موجودیت های گارانتی برند(BrandGuarantee)، گارانتی نوع محصول (BrandProductTypeGuarantee) و گارانتی محصول(ProductGuarantee) که همگی در جدول Guarantee تجمیع شده اند.

مثال : ارث بری درون جدولی مورد سرویس (ServiceItem) که بر اساس نوع سرویس(ServiceTypeEnum) به دو موجودیت سرویس تعمیر(ServiceRepair) و سرویس تست (ServiceTest) تقسیم میشود. روابط و خصوصیات موجودیت مورد سرویس نیز بر حسب اینکه داده سرویس تعمیر است و یا سرویس تست اختصاصی سازی می شوند. موجودیت مفهومی سرویس تعمیر سه رابطه از موجودیت مورد سرویس را با جداول تصاویر سرویس (ServiceItemPartImage) ، مورد تعمیرات (ServiceItemRepair) و درخواست قطعه (RequestProductPart) و همچنین یک ستون به نام مبلغ کل قطعات (RepairPartTotalPrice) را اختصاصی می کند. موجودیت مفهومی سرویس تست نیز به صورت اختصاصی با جدول مورد تست (ServiceItemTest) در ارتباط است. این رابطه ارث بری همپوشان (Overlap) می باشد و امکان اختصاصی نمودن همزمان مورد سرویس به هر دو موجودیت سرویس تست و سرویس تعمیر وجود دارد (با انتخاب خصوصیت نوع سرویس به مقدار 3).

مثال : رابطه بین موجودیت پدر دفتر (Office) و موجودیتهای مفهومی فرزند آژانس (Agency) و دفتر خدماتی (Workshop)

همانطور که اشاره شد، هر رابطه ارث بری مجموعه ای معنی دار از یک یا چند رابطه بین موجودیتهای پدر و فرزند می باشد. هر رابطه بین موجودیت پدر و فرزند در مدل فراداده با دو جهت ایجاد می گردد. در ادامه به مانند انواع روابط قبلی هر جهت رابطه جداگانه بررسی می شود.

### رابطه از موجودیت پدر به فرزند :

|  |
| --- |
| توضیحات |
| * در چینش خودکار فرم با برخورد به اولین رابطه پدر به فرزند، بقیه روابط پدر به فرزند هم گروه نیز در یک گروه رابطه قرار می گیرند و بصورت یکجا نمایش داده می شوند (BizEntityUIComposition.GetOtherRelationshipsGroups: 5d69f05cfdd5) |
| * صرف نظر از رابطه پدر به فرزند برای موجودیت ورود اطلاعات که خود از رابطه فرزند به پدر بوجود آمده و ارث بری IsDisjoint است (BizTableDrivedEntity.GetRelationshipSkipMode: 20a74a0bbfe2) |
| * ایجاد شرط مخفی کردن رابطه در صورتی که رابطه ارث بری مقدار تعیین کننده روابط داشته باشد (UIActionActivityManager.CheckISADeterminerStates: 56344e195b36) |
| * اعتبارسنجی روابط ارث بری در فرمهای ورود اطلاعات (UIValidationManager.ValidateISARelationships: 3f64c5907ac5) |

### رابطه از موجودیت فرزند به پدر :

|  |
| --- |
| توضیحات |
| * روابط فرزند به پدر در ساخت لیست نمایش پیش فرض کاربرد دارند (BizEntityListView.GenereateDefaultListViewColumns: e0b50248b309) |
| * روابط فرزند به پدر در ساخت لیست جستجوی پیش فرض کاربرد دارند (BizEntitySearch.GenereateDefaultSearchColumns: 9e5b4c36a21c) |
| * در ساخت خودکار ظاهر فرم، نوع رابطه فرزند به پدر در جایگذاری رابطه در فرم تاثیر گذار است (BizEntityUIComposition.FKRelationshipHasReservedPosition: 7387bc372bcb) |
| * صرف نظر از رابطه پدر به فرزند برای موجودیت ورود اطلاعات که خود از رابطه فرزند به پدر بوجود آمده و ارث بری IsDisjoint است (BizTableDrivedEntity.GetRelationshipSkipMode: 20a74a0bbfe2) |
| * ایجاد شرط مخفی کردن رابطه در صورتی که رابطه ارث بری مقدار تعیین کننده روابط داشته باشد (UIActionActivityManager.CheckISADeterminerStates: 56344e195b36) |
| * اعتبارسنجی روابط ارث بری در فرمهای ورود اطلاعات (UIValidationManager.ValidateISARelationships: 3f64c5907ac5) |

## روابط اتحاد :

روابط اتحاد نیز تعریف مفهومی و سطح بالاتری از پایگاه داده دارد و به منظور ایجاد اشتراک بین چند موجودیت ماهیتاً متفاوت استفاده می شود. رابطه اتحاد با رابطه ارث بری شباهت ها و تفاوتهایی دارد. از آنجایی که عمدتاً موجودیتهای درگیر در رابطه اتحاد ناهمگون می باشند دیگر از اصطلاح پدر و فرزند استفاده نکرده و به جای آن از عبارت موجودیت اصلی و موجودیت فرعی استفاده می شود. از منظر تعریف رابطه، یک رابطه اتحاد به مانند رابطه ارث بری است که همپوشان (Overlap) نباشد. یعنی هر موجودیت اصلی نماینده تنها یک موجودیت فرعی می باشد. تفاوت دیگر در پیاده سازی رابطه می باشند به ترتیبی که جای کلید های اصلی و خارجی در روابط فیزیکی عوض می شود و کلیدهای خارجی روابط به جای موجودیت فرعی در همان موجودیت اصلی نگهداری می شوند. به این ترتیب کلید اصلی موجودیت فرعی اتحاد به عنوان کلید خارجی در موجودیت اصلی اتحاد قرار می گیرد. موجودیت اصلی نیز دارای کلید اصلی مجزا و یکتای خود می باشد. این روش به این دلیل است که اولاً عمدتاً ارتباط معنایی یکسانی بین کلید های موجودیت های درگیر وجود ندارد و همچنین با تجمیع کلیدهای موجودیتهای فرعی در موجودیت اصلی امکان کنترل فیزیکی همپوشان نبودن رابطه تسهیل می شود. تفاوت دوم رابطه اتحاد در مفهوم شرکت پذیری رابطه (TotalParticipation) می باشد. در رابطه ارث بری شرکت پذیری رابطه به این معنی بود که آیا هر موجودیت پدر باید با موجودیت فرزندی در ارتباط باشد یا خیر(طبیعتاً تمامی موجودیتهای فرزند نیز با پدر در ارتباط بودند). در رابطه اتحاد شرکت عمومی رابطه به این معنی است که آیا هر موجودیت فرعی باید با موجودیت اصلی در ارتباط باشد یا خیر(طبیعتاً تمامی موجودیتهای اصلی تنها با یک موجودیت فرعی در ارتباطند).

کاربر می تواند روابطه اتحاد را چه در فرم استخراج روابط(frmImportRelationships.DetermineRelationshipType: 390db5fb494d) و چه پس از استخراج نیز در برنامه مدیریت فراداده تعریف و مدیریت نماید (frmConceptualRelationships: fdf71cf178e5).

مثال: ارتباط اتحاد بین موجودیتهای فرعی خدمات اضافه (ServiceAdditionalItem) و مورد سرویس (ServiceItem) و موجودیت اصلی مورد خلاصه سرویس (ServiceConclusionItem)

### رابطه اتحاد از اصلی به فرعی :

|  |
| --- |
| توضیحات |
| * در ساخت خودکار ظاهر فرم، نوع رابطه فرزند به پدر در جایگذاری رابطه در فرم تاثیر گذار است (BizEntityUIComposition.FKRelationshipHasReservedPosition: 7387bc372bcb) |
| * صرف نظر از رابطه اصلی به فرعی اتحاد برای موجودیت ورود اطلاعات که خود از رابطه فرعی به اصلی بوجود آمده (BizTableDrivedEntity.GetRelationshipSkipMode: 20a74a0bbfe2) |
| * ایجاد شرط مخفی کردن رابطه در صورتی که رابطه اتحاد مقدار تعیین کننده روابط داشته باشد (UIActionActivityManager.CheckUnionDeterminerStates: 0199f3874e7c) |
| * اعتبارسنجی روابط اتحاد در فرمهای ورود اطلاعات (UIValidationManager.ValidateUnionRelationships: e50167077e44) |

### رابطه اتحاد از فرعی به اصلی:

|  |
| --- |
| توضیحات |
| * صرف نظر از رابطه پدر به فرزند برای موجودیت ورود اطلاعات که خود از رابطه فرزند به پدر بوجود آمده و ارث بری IsDisjoint است (BizTableDrivedEntity.GetRelationshipSkipMode: 20a74a0bbfe2) |
| * ایجاد شرط مخفی کردن رابطه در صورتی که رابطه اتحاد مقدار تعیین کننده روابط داشته باشد (UIActionActivityManager.CheckISADeterminerStates: 56344e195b36) |
| * اعتبارسنجی روابط اتحاد در فرمهای ورود اطلاعات (UIValidationManager.ValidateISARelationships: 3f64c5907ac5) |

## روابط چند به چند : ؟؟؟

## ایجاد یک رابطه جدید:

یکی از قابلیتهای برنامه مدیریت فراداده، امکان تعریف یک رابطه بین موجودیتها علاوه بر روابط فیزیکی حاکم بر پایگاه داده هدف می باشد (frmRelationshipCreate: cb74ea91b9d3). پس از تعریف یک رابطه جدید، آن رابطه تمام مشخصات و خصوصیات روابط را به مانند روابط فیزیکی پایگاه داده خواهد داشت.

## تبدیل روابط به یکدیگر :

همانگونه که ذکر شد، فرم استخراج روابط تنها یک مرحله ابتدایی در تعریف روابط مدل فراداده می باشد. در ادامه و در برنامه مدیریت فراداده می توان انواع روابط استخراج شده را تبدیل و بازتعریف نمود. البته تبدیل هر نوع رابطه به دیگری ممکن است بنابر قوانین و داده های پایگاه داده هدف دارای پیش شرطها و محدودیتهایی باشد که پیش از تبدیل نوع رابطه کنترل می گردد (BizRelationshipConverter.GetRelationshipConvertOptions: 6c1571817afe و BizRelationshipConverter.ConvertRelationship: abce7e19b235).

# درخت منو

درخت منو در واقع درختی از مجوعه عناوین فرمها، گزارشها ومنابعی است که در برنامه ورود اطلاعات مورد استفاده قرار می گیرند. در این بخش به موجودیت درخت منو در مدل فراداده و نحوه تعریف و کاربردهای آن پرداخته می شود.

## تعریف درخت :

درخت منو در یک فرم مخصوص تعریف می شود(frmNavigationTree: 16800307f2f0). در این فرم ساختار درختی پایگاه داده های موجود در مدل فراداده مبنای ساخت درخت منو قرار می گیرد((BizDatabaseToObject.GetDatabaseChildObjects: c1e9819478c9).). کاربر می تواند با کشیدن منابع یک یا چند پایگاه داده به درخت منو و یا به صورت دستی به اصلاح درخت منو بپردازد. در واقع درخت منو می تواند ترکیبی از شاخه های تودرتو باشد که هر شاخه خصوصیات ظاهری و مفهومی خود را دارد. این شاخه ها را می توان با درگ/دراپ ساختارهای پایگاه داده هدف و یا بصورت دستی ایجاد نمود. بطور کلی هر شاخه منو دارای ساختار زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Name | نام شاخه که از نام موجودیت پشت آن برگرفته می شود. |
| ParentItem | شاخه پدر |
| ObjectCategory | نوع موجودیت پشت شاخه |
| TableDrivedEntityID | شناسه موجودیت کسب و کار مرتبط |
| Tooltip | توضیحات اضافی نمایش شاخه |

### انواع شاخه قابل تعریف (ObjectCategory):

انواع شاخه قابل تعریف و خصوصیات آنها در جدول زیر گردآوری شده است(frmNavigationTree.GetNodeHeader: ed015b6c86a7).

|  |  |
| --- | --- |
| انواع درخت منو | توضیحات |
| فولدر | در صورت درگ/دراپ مفاهیمی مانند Database و Schema به نوع فولدر تبدیل می شوند، همچنین بصورت دستی توسط کاربر قابل تعریف می باشند. |
| فرم ورود اطلاعات | فرم ورود اطلاعات هر موجودیت |
| گزارش موجودیت | گزارشهای تعریف شده برای یک موجودیت، البته هر گزارش می تواند خود از انواع مختلفی باشد که درجای خود به آنها پرداخته خواهد شد. |
| آرشیو موجودیت | فرم آرشیو موجودیت |
| نامه های موجودیت | فرم نامه های موجودیت |
| نمای داده موجودیت | فرم نمای داده موجودیت |
| گرید داده موجودیت | فرم گرید داده موجودیت |

لازم به ذکر است که فرم تعریف درخت منو در زمان استخراج پایگاه داده هدف نیز کاربرد دارد. در آنجا هم یا میانبری به همین فرم می توان ساختار اولیه درخت منوی برنامه را تعریف نمود.

## کاربرد درخت منو :

کاربرد درخت منو در برنامه ورود اطلاعات می باشد. این درخت در پنلی نمایش داده شده تا کاربر بتواند به سهولت بین فرمها و گزارشات دارای دسترسی پیمایش کند(AgentUICoreMediator.ShowNavigationTree: 39c7547dd08c). البته این منو با توجه به دسترسی های کاربر جاری خوانده شده و نمایش داده می شود. دسترسیهای هر شاخه وابسته به مشخصات کاربر جاری، نوع شاخه و موجودیت پشت هر شاخه است (BizNavigationTree.GetNavigationTree: 807e75a4ae6). با کلیک به روی هر شاخه منو رویداد NavigationMenuClicked فراخوانی می شود که بنابر نوع شاخه فرمی را در برنامه نمایش می دهد(AgentUICoreMediator.NavigationMenuClicked: a3a9f1a1fdd4)

پنل منوی ذکر شده در برنامه ورود اطلاعات در دو صورت دریافت و بروزرسانی می شود. اول بعد از ورود کاربر به سامانه و نمایش فرم اصلی برنامه (AgentUICoreMediator.LoginForm\_LoginRequested: 99b51194b636) و دوم در صورت تغییر مشخصات سازمانی و امنیتی کاربر جاری در فرم مربوطه در برنامه ورود اطلاعات (AgentUICoreMediator.SecuritySettings\_Clicked: 3c8d2871a4e2).

# فرم کار با موجودیت ورود اطلاعات

اساساً هدف اصلی از بسیاری تعاریف انجام شده در مدل فراداده نمایش فرمهای کار با موجودیت و داده های آن می باشد. به عبارت دیگر در یک فرم ورود اطلاعات شرایط برای انجام هر چه موثرتر عملیات های CRUD فراهم می باشد. یک فرم ورود اطلاعات شامل ستونها و روابط موجودیت و همچنین دکمه و میانبرهایی به منظور تعیین و انجام عملیات های جستجو، ثبت و .. می باشد.

برای نمایش هر فرم ورود اطلاعات مراحل زیر انجام می شود :

ابتدا یک EditEntityAreaInitializer ساخته می شود(EditEntityAreaInitializer: 7bb9feae0514). در این شیء مشخصات اولیه ایجاد فرم ذکر شده است. پارامترهای آن در جدول زیر آمده است:

|  |  |
| --- | --- |
| بخش برنامه | توضیحات |
| EntityID | شناسه موجودیت |
| DataMode | نحوه نمایش فرم از نظر تک داده و یا چندتای می باشد |
| IntracionMode | نحوه تعامل با فرم |
| TailDataValidation | رشته رابطه اعتبارسنجی؟؟ |
| Preview | تولید نمونه برای تنظیمات نمایش فرم |

### خصوصیت IntracionMode :

یکی از خصوصیات مهم هر EditEntityAreaInitializer،خصوصیت IntracionMode می باشد که دارای پنج مقدار است :

|  |  |
| --- | --- |
| نام خصوصیت | توضیحات |
| CreateDirect | فرم مستقیم برای ورود اطلاعات |
| CreateSelectDirect | فرم مستقیم برای ورود اطلاعات با قابلیت انتخاب |
| CreateInDirect | فرم غیر مستقیم ورود اطلاعات |
| CreateSelectInDirect | فرم غیر مستقیم ورود اطلاعات و با انتخاب |
| Select | فقط انتخاب داده |

یک رابطه در فرم های ورود اطلاعات، در واقع ارتباط موجودیت جاری با موجودیتی دیگر می باشد. بنابراین برای موجودیت دیگر نیز یک فرم مورد نیاز است. حال این فرم می تواند بصورت مستقیم در فرم میزبان قرار گیرد یا می توان بصورت یک میانبر باشد که فرم موجودیت رابطه را در پنجره ای در فرم میزبان باز می کند. اساساً این فرم رابطه می تواند خود قابلیت ورود اطلاعات داشته باشد یا فقط هدف انتخاب یک داده برای ایجاد رابطه باشد. این نحوه تعامل با فرمها در خصوصیت IntracionMode اشاره می شود. عبارت Direct و InDirect در این متغیر نشان دهنده نوع نمایش مستقیم و یا غیر مستقیم (میانبری) فرم است. عبارت Create نشان دهنده قابلیت اصلاح داده در فرم و عبارت Select نشان دهنده قابلیت انتخاب داده می باشد.

### خصوصیت DataMode :

DataMode خصوصیت مهم دیگری است که تعیین کننده نوع فرم است. فرمهای ورود اطلاعات بصورت کلی به دو نوع تک داده (EditEntityAreaOneData) و یا چند داده ای (EditEntityAreaMultipleData) تقسیم می شوند. به دلیل جدابودن بسیاری از دغدغه های پیاده سازی در این دو نوع فرم، کلاسهای مربوطه بصورت جداگانه پیاده سازی شده اند. البته اشتراکات زیادی در عملکرد این دو نوع فرم وجود دارد که از طریق ارث بری از BaseEditEntityArea پوشش داده شده است.

در ادامه بحث نمایش فرم ورود اطلاعات، EditEntityAreaInitializer تولید شده به متد GetEditEntityArea فرستاده می شود(BaseEditEntityArea.GetEditEntityArea: 9ad46f1525db). هدف از این متد ایجاد یک I\_EditEntityArea یا همان خروجی انتزاعی یک فرم ورود اطلاعات می باشد. مهمترین اتفاق هایی که در این متد رخ می دهد، کنترل دسترسیهای کاربر به فرم ورود اطلاعات و نهایتاً تولید ظاهر اولیۀ فرم بوسیله فراخوانی روال SetAreaInitializerAndGenerateView فرم می باشد(BaseEditEntityArea.SetAreaInitializerAndGenerateView: 079c6e89e544). در این متد نیز بسته به نوع پارامتر IntracionMode ارسال شده و مستقیم و یا غیر مستقیم بودن فرم ظاهر اولیه فرم شامل کنترلهای خصوصیات و روابط و یا دکمه های فرم ساخته می شوند. ابتدا به حالت ایجاد ظاهر اولیه فرم مستقیم ورود اطلاعات پرداخته می شود.

## **ساخت ظاهر فرم مستقیم (GenerateDataView):**

اگر InteractionMode اولیه فرم یکی از حالات CreateDirect یا CreateSelectDirect باشد تنها ظاهری که از فرم ساخته می شود همان نمای مستقیم ورود اطلاعات بصورت کامل می باشد. برای این امر متد GenerateDataView فراخوانی می شود که در این متد قابل کلی فرم، کنترلهای ورود اطلاعات، کنترلهای روابط و دکمه ها ساخته می شوند. در اولین گام بر حسب تک دادهو یا چند داده بودن فرم مستقیم ورود اطلاعات قالب واسط کاربری فرم ساخته می شود. ( UIManager.GenerateEditEntityAreaOneDataView: 7c2a866deda1 و UIManager.GenerateEditEntityAreaOneDataView: 346253bf3bf5). گام بعدی ساخت دکمه های فرم می باشد.

### ساخت دکمه ها برای فرم ورود اطلاعات مستقیم :

از طریق این دکمه ها عملیات های گوناگون در فرم تقسیم بندی و اجرا می گردند. ساخت دکمه ها در متد GenerateCommands انجام می شود(BaseEditEntityArea.GenerateCommands: ada6635d8043). دکمه های فرم مستقیم اساساً به دو دسته تقسیم می گردند:

* دکمه های عملیات CRUD
* دکمه های قابلیتهای وابسته به داده انتخاب شده در فرم

#### دکمه های عملیات CRUD:

|  |  |  |
| --- | --- | --- |
| نام | توضیحات | شرط عمومی نمایش |
| ClearCommand | خالی کردن فرم از داده (ClearCommand: d587de1402da) | بدون شرط |
| SaveCommand | ثبت داده در پایگاه داده (SaveCommand: 21d90a42267d) | برای فرم اصلی فقط نمایش داده می شود (و نه فرمهایی که از رابطه نمایش داده می شوند) |
| DeleteCommand | حذف داده از پایگاه داده (DeleteCommand: a29b9e9cf377) | برای فرم اصلی فقط نمایش داده می شود |
| SearchCommand | جستجوی داده از پایگاه داده (SearchCommand: cda8ae466b21) | برای فرمهایی که InteractionMode آنها قابلیت انتخاب داده داشته باشد |
| UpdateAndCloseDialogCommand | کنترل داده ها و بستن پنجره فرم ورود اطلاعات (UpdateAndCloseDialogCommand: dc6ea404ac9d) | برای فرمهایی که InteractionMode آنها قابلیت باز شدن غیر مستقیم دارد |
| AddCommand | افزودن داده به جدول داده ها در فرم های چند داده ای (AddCommand: 5168bfcab87f) | برای صرفاً فرمهای چند داده ای |
| RemoveCommand | حذف داده از جدول داده ها در فرم های چند داده ای (RemoveCommand: 43c89dd5b4bf) | برای صرفاً فرمهای چند داده ای |

#### دکمه های قابلیتهای وابسته به داده انتخاب شده در فرم:

|  |  |  |
| --- | --- | --- |
| نام | توضیحات | شرط نمایش |
| InfoCommand | اطلاعات سیستمی درباره داده انتخاب شده (InfoCommand: 83c8ff061a3d) |  |
| LogReportCommand | گزارش لاگ درباره داده انتخاب شده (LogReportCommand: ffc2a3c7a92f) |  |
| ArchiveCommand | نمایش فرم آرشیو برای داده انتخاب شده (ArchiveCommand: dc80ddb6a7f3) | در صورت وجود دسترسی نمایش یا اصلاح آرشیو برای کاربر |
| LetterCommand | نمایش فرم نامه برای داده انتخاب شده (LetterCommand: ae44e26fef6b) | در صورت وجود دسترسی نمایش یا اصلاح نامه برای کاربر |
| DataViewCommand | نمایش گرافیکی داده انتخاب شده در فرم (DataViewCommand: 939641817cea) | برای فرم اصلی فقط نمایش داده می شود (و نه فرمهایی که از رابطه نمایش داده می شوند) |
| GridViewCommand | نمایش جدولی داده های فرم (GridViewCommand: a7111982b91c) | برای فرم اصلی فقط نمایش داده می شود (و نه فرمهایی که از رابطه نمایش داده می شوند) |
| DataLinkCommand | نمایش فرم لینک داده برای داده انتخاب شده (DataLinkCommand: 3de1bf09614c) | برای فرم اصلی فقط نمایش داده می شود (و نه فرمهایی که از رابطه نمایش داده می شوند) |
| GraphCommand | نمایش فرم گراف داده برای داده انتخاب شده (GraphCommand: 6ef5fd4bbae5) | برای فرم اصلی فقط نمایش داده می شود (و نه فرمهایی که از رابطه نمایش داده می شوند) |
| EntityCommand | دکمه های داینامیک تعریف شده برای یک موجودیت که کدی اختصاصی و از پیش تعریف شده را با ارسال داده انتخاب شده اجرا می کنند (EntityCommand: 1c0a26563d22) |  |

به منظور حفظ پیوستگی در ارائه مطالب، در بخش های مربوطه دیگر به تشریح عملکرد و شروط هر کدام از نوع دکمه های ذکر شده پرداخته خواهد شد. در ادامه پس از قالب فرم و ساخت دکمه ها نوبت ساخت کنترلهای ورود اطلاعات فرم می باشد.

اولین گام در ساخت کنترلهای ورود اطلاعات، ساخت الگوی قرار گیری کنترل ها می باشد. این موارد می تواند شامل کنترلهای ساده به مانند تکست باکس و یا یک رابطه که خود می تواند یک فرم مستقیم برای موجودیت دیگر، یک میابنر به فرم موجودیت دیگر و یا یک میانبر برای انتخاب داده های موجودیت دیگر باشد. هر کدام از این کنترلها نیز می توانند دارای جزئیات ظاهری و خصوصیاتی باشند. دلیل ساخت چنین الگویی برای فرم ورود اطلاعات اینست که از پیچیدگی و وابستگی منطق نمایش فرمهای ورود اطلاعات با منطق عملکردی موجودیتها و روابط تا حد امکان جلوگیری شود.

نکته : زمانی که یک رابطه خود چه مستقیم و یا غیر مستقیم (از طریق میانبر) به فرمی ورود اطلاعات منتهی می شود، تقریباً اکثر موارد مطرح شده در این بخش یعنی ساخت ظاهر فرم مستقیم نیز در مورد فرم آن رابطه نیز صدق می کند. به عبارت دیگر اکثر رویدادهای ذکر شده نیز به صورت مجزا برای آن فرم اجرا می شود. حال اگر فرم مستقیم بود ظاهر فرم همزمان با ظاهر فرم پدر ساخته می شود و اگر غیر مستقیم بود پس از اولین بار باز نمودن فرم ورود اطلاعات از طریق میانبر.

در ادامه نحوه ساخت و دریافت الگو فرمهای ورود اطلاعات بررسی می شود.

### دریافت موجودیت الگوی فرمهای ورود اطلاعات:

هر فرم ورود اطلاعات برای دریافت الگوی خود به سراغ متغیر DataEntryEntity (BaseEditEntityArea.DataEntryEntity: fc084e9c81fc) می رود. برای فرم اصلی این الگو از متد GetDataEntryEntity(BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7) دریافت می شود. این الگو ورود اطلاعات شامل جزئیات فرم جاری و همچنین الگوهای تمامی فرمهای روابط مستقیم در فرم (در صورت وجود) می باشد. بنابراین برای تمامی فرمهای ورود اطلاعات روابط مستقیم که الگوی آنها همراه با موجودیت پدر آمده است دیگر متغیر DataEntryEntity سراغ GetDataEntryEntity نمی رود و از همان الگوی پیشتر فرستاده شده توسط روابط پدر استفاده می شود. برای فرمهای غیر مستقیم که دارای فرم ورود اطلاعات می باشند نیز هر بار که از طریق میانبر برای اولین بار فرم ورود اطلاعات مربوطه باز شد طبق روال ذکر شده متغیر DataEntryEntity به سراغ متد GetDataEntryEntity و دریافت الگوی فرم ورود اطلاعات می رود.

الگوی فرم ورود اطلاعات خود یک موجودیت سیستمی از نوع DataEntryEntityDTO می باشد که ساختار آن در زیر آمده است :

|  |  |  |
| --- | --- | --- |
| نام خصوصیت | نوع | توضیحات |
| IsReadonly | bool | آیا موجودیت فقط خواندنی است |
| Columns | List<ColumnDTO> | ستونهای ورود اطلاعات |
| Relationships | List<DataEntryRelationshipDTO> | روابط ورود اطلاعات که خود از نوع مخصوص DataEntryRelationshipDTO می باشند |
| ParentDataEntryRelationship | DataEntryRelationshipDTO | رابطه پدر |
| SkippedRelationships | List<RelationshipDTO> | روابط صرف نظر شده |
| Info | string |  |
| HasNotDeleteAccess | bool | آیا دسترسی حذف داده وجود دارد |
| UICompositions | EntityUICompositionDTO | الگوی چینش و ظاهر کنترلها |

در ادامه روند تهیه DataEntryEntityDTO در متد GetDataEntryEntity را بررسی می کنیم .

در گام اول با استفاده از متد CheckDataEntryPermission (BizTableDrivedEntity.CheckDataEntryPermission: 8be394c313b9) دسترسی های عمومی و اختصاصی به روی موجودیت ها، ستونها و روابط کنترل می شود. در واقع خروجی این متد یک TableDrivedEntityDTO است که ستونها و روابط بدون دسترسی از آن حذف شده اند و همچنین دسترسیهای فقط خواندنی به روی موجودیت، ستونها و روابط تعیین شده است. سه خصوصیت خروجی اصلی متد GetDataEntryEntity یعنی IsReadonly، HasNotDeleteAccess و Columns نیز عیناً از موجودیت TableDrivedEntityDTO اشاره شده بدست می آیند. امّا بررسی کنیم در CheckDataEntryPermission این سه خصوصیت چگونه تعیین می شوند.

#### خصوصیت IsReadonly :

اینکه آیا موجودیت فقط خواندنی است یا خیر از دو مرجع تعیین می شود. اول اینکه در برنامه مدیریت فراداده فقط خواندنی تعیین شده باشد. دوم اینکه کاربر جاری دسترسی فقط خواندنی به موجودیت داشته باشد(BizTableDrivedEntity.EntityIsReadonly: a8c7523b7591).

#### خصوصیت HasNotDeleteAccess:

اینکه آیا امکان حذف برای داده های موجودیت در فرم ورود اطلاعات وجود ندارد. این مورد از دو شرط تعیین می شود اول فقط خواندنی نبودن موجودیت (در بند قبل) و داشتن دسترسی اصلاح و حذف توسط کاربر بروی موجودیت (BizTableDrivedEntity.DataHasNotDeleteAccess: f7b87aadf83b)

#### خصوصیت Columns :

تمامی ستونهایی که در فرم ورود اطلاعات قابل نمایش باشند به همراه تعیین وضعیت فقط خواندنی بودن یا نبودن. حال سوال اینست که چه ستونهایی قابل نمایش نمی باشند، در واقع ستونهایی که کلید اصلی نباشند و :

* بر روی آنها رابطه وجود داشته باشد و رابطه به هر دلیلی غیرقابل نمایش باشد. (در واقع ستون کلید خارجی باشد و رابطه حذفی باشد)
* کاربر به آنها دسترسی نداشته باشد. یعنی یا ستون غیرفعال باشد یا کاربر دسترسی امنیتی به کل نداشته باشد یا ستون غیر قابل ورود اطلاعات باشد (BizColumn.DataIsAccessable: 0c6adb5b0bdc).

امّا تعیین شرط فقط خواندنی بودن ستون. شرایط زیر برای تعیین فقط خواندنی بودن ستون وجود دارد:

* اگر موجودیت فقط خواندنی باشد تمام ستونها فقط خواندنی می شوند، همچنین اگر ستون در برنامه مدیریت داده فقط خواندنی باشد ، اگر ستون شناسه خودکار باشد (Identity)، اگر کاربر دسترسی فقط خواندنی به ستون داشته باشد (BizColumn.DataIsReadonly: 46ee61f89778)
* اگر فرم از رابطه اصلی به فرعی بوجود آمده باشد و ستون کلید خارجی آن رابطه باشد (BizTableDrivedEntity.CheckDataEntryPermission: 8be394c313b9) حذف شد

در مورد روابط علاوه بر بررسی دسترسی های مشابه موارد ذکر شده در متد CheckDataEntryPermission، در متد اصلی یا همان GetDataEntryEntity پردازش های دیگری نیز انجام می گردد. این پردازشها شروط دیگری را به منظور نمایش یا عدم نمایش رابطه بررسی می کند. همچنین خصوصیات دیگری مانند نوع تعامل با رابطه برای هر رابطه تعیین می شود. به این منظور برای هر رابطه، موجودیت DataEntryRelationshipDTO در خروجی الگوی نهایی فرم ورود اطلاعات باز تولید می گردد.

#### خصوصیت Relationships:

ابتدا نحوه تعیین دستریهای روابط در همان متد CheckDataEntryPermission بررسی می شود.

رابطه هایی که دارای شرایط زیر باشند به معنی عدم دسترسی کاربر در فرم ورود اطلاعات می باشد و بنابراین از لیست روابط حذف می گردند.

* همانطور که پیشتر نیز ذکر شد دسترسی های مستقیم روابط از روی دسترسی های ستون های کلید خارجی تعیین می گردد. بنابراین رابطه برای ورود اطلاعات حذف می گردد در صورتی که کاربر به ستونهای کلید خارجی رابطه دسترسی ورود اطلاعات نداشته باشد. یعنی یا ستون غیرفعال باشد یا کاربر دسترسی امنیتی به کل نداشته باشد یا ستون غیر قابل ورود اطلاعات باشد. (BizRelationship.DataIsAccessable: d2ac62182cab).
* کاربر به موجودیت طرف دوم دسترسی نداشته باشد. یعنی یا موجودیت غیرفعال باشد یا کاربر دسترسی نداشته باشد (BizRelationship.DataIsAccessable: d2ac62182cab)

همانطور که پیشتر ذکر شد، شرط فقط خواندنی بودن رابطه نیز در متد CheckDataEntryPermission کنترل می شود. شرایط زیر برای تعیین فقط خواندنی بودن روابط وجود دارد:

* اگر ستون کلید خارجی فقط خواندنی باشد (طبق قوانین ستون فقط خواندنی در بخش قبل) (BizRelationship.DataIsReadonly: acae9fe7903b)

پس از انجام متد CheckDataEntryPermission و تعیین شدن دسترسیهای لازم به روی ستونها و روابط، به متد GetDataEntryEntity برگشته و باز هم فیلترهایی به روی روابط موجودیت مورد ورود اطلاعات انجام می گردد. این فیلترهای رابطه به علت قوانین سیستمی سامانه ورود اطلاعات می باشد و باعث شکیل تر شدن و کاربردی تر شدن واسط کاربری فرم ورود اطلاعات می گردد. در واقع به علت اینکه در فرم ورود اطلاعات یک موجودیت روابط، پای سایر موجودیت ها و فرمهای دیگر را به میان می کشند، دغدغه نحوه نمایش روابط موجودیت با موجودیتهای و عدم پیچیدگی فرم در نمایش این روابط تودرتو می باشد. اینکه آیا همه روابط نمایش داده شوند، آیا روابط بصورت مستقیم نمایش داده شوند یا غیر مستقیم و آیا هر رابطه انتخابی است یا ورود اطلاعاتی یا هر دو؟ اینکه روابط موجودیت های طرف دیگر روابط نمایش داده شوند و اگر بله نحوه نمایش به چه صورت باشد. به این منظور برای ها رابطه یک موجودیت جدید از نوع DataEntryRelationshipDTO ساخته می شود که این خصوصیات تکمیلی به آن ضمیمه شده است. این نوع اشاره شده دارای ساختار زیر است :

|  |  |  |
| --- | --- | --- |
| نام خصوصیت | نوع | توضیحات |
| TargetDataEntryEntity | DataEntryEntityDTO | موجودیت ورود اطلاعات مربوطه |
| Relationship | RelationshipDTO | رابطه مربوطه |
| DataMode | DataMode | نوع داده ها تکی یا چندتایی |
| IntracionMode | IntracionMode | نوع تعامل با فرم |
|  |  |  |

**خصوصیت DataMode :**

همانطور که در توضیحات EditEntityAreaInitializer در بخش ایجاد فرم ورود اطلاعات توضیح داده شده این خصوصیت تعیین کننده چند داده ای بودن فرم ها می باشد. اگر رابطه یک به چند باشد مقدار آن Multiple و در غیر اینصورت One مقداردهی می شود. دقیقاً از این خصوصیت بر روی هر DataEntryRelationshipDTO برای ساختن EditEntityAreaInitializer فرم مربوطه استفاده می گردد. (BaseEditEntityArea.GenerateRelationshipControlEditArea: 7b5673de233c)

**خصوصیت IntracionMode :**

درباره این نوع خصوصیت نیز در توضیحات EditEntityAreaInitializer در بخش ایجاد فرم ورود اطلاعات توضیح داده شده است. نحوه تعیین این خصوصیت برای هر رابطه در فرمهای ورود اطلاعات به شرح زیر است (BaseEditEntityArea.GenerateRelationshipControlEditArea: 7b5673de233c) :

* اگر رابطه پدر وجود داشته باشد و رابطه پدر خود یک به چند باشد کلا ورود اطلاعات مستقیم روابط فرم جاری ممکن نمی باشد.
* اگر طرف دیگر رابطه قابل ورود اطلاعات باشد (IsOtherSideCreatable == true)،
  + اگر قابل ورود اطلاعات مستقیم باشد و در سلسله روابط پدر، موجودیت طرف دوم به طور مستقیم ورود اطلاعات نشده باشد
    - اگر یک به چند باشد حالت CreateDirect
    - در غیر اینصورت CreateSelectDirect
  + اگر ورود اطلاعات مستقیم نباشد یا در سلسله روابط مستقیم پدر، موجودیت طرف دوم به طور مستقیم ورود اطلاعات شده باشد
    - اگر یک به چند باشد حالت CreateInDirect
    - در غیر اینصورت CreateSelectInDirect
* اگر طرف دیگر رابطه قابل ورود اطلاعات نباشد (صرفاً انتخابی باشد)
  + - اگر یک به چند باشد از رابطه صرف نظر می شود
    - اگر یک به چند نباشد حالت Select

فیلتر دیگری که بروی روابط انجام می شود، صرف نظر از برخی از روابط برای فرمهایی که خود از رابطه بوجود آمده اند می باشد. در واقع به منظور جلوگیری از پیچیدگی تودرتوی فرمها و روابط، از برخی از روابط غیر ضروری و درجه دو صرف نظر می گردد. به منظور نگهداری سوابق، هر رابطه حذف شده به خصوصیت SkippedRelationships موجودیت ورود اطلاعات آن فرم افزوده خواهد شد. امّا قوانین صرف نظر از روابط به دو قسم تعیین می گردند. ابتدا قوانین صرف نظر از روابطی که نوع تعامل با آنها اهمیتی ندارد و قانون حذف کلی است(BizTableDrivedEntity.GetRelationshipSkipMode: 20a74a0bbfe2) :

* + - همانطور که پیشتر ذکر شد روابط در مدل فراداده دوطرفه می باشند. بنابراین اگر فرم B از رابطه موجودیت A به B به نام X بوجود آمده باشد، طبیعتاً از بین روابط B، از رابطه عکس X صرف نظر می شود
    - اگر رابطه بالاتر ارث بری فرزند به پدر و IsDisjoint باشد و رابطه جاری پدر به فرزند همان رابطه ارث بری باشد از رابطه جاری صرف نظر می شود (مثلا پدر شخص حقیقی به شخص باشد و این رابطه شخص به شخص حقوقی باشد)
    - اگر رابطه بالاتر زیر اتحاد به اتحاد باشد و رابطه جاری اتحاد به زیر اتحاد همان رابطه اتحاد باشد از رابطه جاری صرف نظر می شود.

قوانین صرف نظر از برخی روابط نیز وابسته به نوع تعامل با فرم ورود اطلاعات آن رابطه می باشد (BizTableDrivedEntity.ShouldSkipRelationshipDepended: 57991005621a) :

* + - اگر رابطه بالاتر مستقیم باشد، اگر موجودیت جاری فرم ورود اطلاعات مستقل داشته باشد، اگر رابطه جاری غیر قابل صرف نظر نباشد و اجباری نباشد از رابطه جاری صرف نظر می شود.

**خصوصیت TargetDataEntryEntity**

همانطور که پیشتر ذکر شد طرف دیگر هر رابطه خود یک فرم ورود اطلاعات در نظر گرفته می شود. حال اگر رابطه مستقیم (IntracionMode از نوع مستقیم باشد) باشد موجودیت ورود اطلاعات آن رابطه نیز از ابتدا تولید شده و به همراه رابطه ارسال می گردد(BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7). طبیعتاً این خصوصیت نیز از نوع DataEntryEntityDTO می باشد. اگر رابطه مستقیم نباشد این خصوصیت و با تاخیر و در صورت فراخوانی فرم ورود اطلاعات مربوطه برای رابطه در واسط کاربری مقداردهی می گردد (BaseEditEntityArea.DataEntryEntity: fc084e9c81fc).

پس از بررسی روابط در خروجی موجودیت ورود اطلاعات به سراغ آخرین خصوصیت موجودیت ورود اطلاعات می رویم.

#### خصوصیت UICompositions:

حال که تمامی ستونها و روابط موجودیت ورود اطلاعات به همراه دسترسیها و نحوه تعامل با آنها مشخص شده اند، تنها مورد باقی مانده نحوه چینش و قرارگیری این عناصر در فرم می باشد. به این منظور، به همراه موجودیت ورود اطلاعات، الگوی ظاهری فرمهای ورود اطلاعات نیز دریافت می شود که این خصوصیت به نام UICompositions و از نوع EntityUICompositionDTO می باشد. یک فرم ورود اطلاعات می تواند شامل مجموعه ای از عناصر به صورت تودرتو باشد، بنابراین ساختار موجودیت EntityUICompositionDTO که عینیتی از اقلام مورد انتظار فرمها می باشد نیز ساختاری درختی و تودرتو دارد. به واقع هر EntityUICompositionDTO نماینده یک قلم ظاهری در فرم می باشد که این قلم می تواند یک پنل، یک TabPage، یک کنترل به مانند یک تکست باکس برای یک ستون یا یک زیر فرم برای یک رابطه باشد. ساختار EntityUICompositionDTO در زیر نمایش داده شده است.

|  |  |  |
| --- | --- | --- |
| عنوان | نوع | توضیحات |
| ObjectCategory | DatabaseObjectCategory | نوع قلم در فرم |
| Title | string | عنوان آن قلم |
| ObjectIdentity | string | شناسه فیلد یا رابطه مرتبط با قلم |
| ParentItem | EntityUICompositionDTO | قلم پدر |
| ChildItems | List<EntityUICompositionDTO> | قلم های فرزند |
| Position | int | ترتیب قلم |
| EntityUISetting | EntityUISettingDTO | مشخصات ظاهری فرم اصلی در صورتی که قلم از نوع فرم اصلی ورود اطلاعات باشد |
| GroupUISetting | GroupUISettingDTO | مشخصات ظاهری پنل در صورتی که قلم از نوع پنل گروه بندی باشد |
| TabGroupUISetting | TabGroupUISettingDTO | مشخصات ظاهری کنترل تب در صورتی که قلم از نوع کنترل تب باشد |
| TabPageUISetting | TabPageUISettingDTO | مشخصات ظاهری برگه تب در صورتی که قلم از نوع برگه تب باشد |
| RelationshipUISetting | RelationshipUISettingDTO | مشخصات ظاهری رابطه و فرم مربوطه در صورتی که قلم از نوع رابطه باشد |
| ColumnUISetting | ColumnUISettingDTO | مشخصات ظاهری کنترل ستون در صورتی که قلم از نوع ستون باشد |
| EmptySpaceUISetting | EmptySpaceUISettingDTO | مشخصات ظاهری فضای خالی در صورتی که قلم از نوع فضای خالی باشد |
| Column | ColumnDTO | اطلاعات ستون مربوطه، در صورتی که قلم از نوع ستون باشد |
| Relationship | DataEntryRelationshipDTO | اطلاعات رابطه مربوطه، در صورتی که قلم از نوع رابطه باشد |

اساساً در دسته بندی اقلام در قالب این موجودیت سه نوع مفهوم متصور است: کانتینرها، کنترلها و فضای خالی.

**کانتینرها :** همان فضاهای دسته بندی شده در فرم هستند که به منظور گروهبندی و مرتب سازی اقلام فرمها استفاده می شوند، به مانند پنل ها و تب ها. هر کانتینر می تواند شامل یک یا چند قلم از هر 3 نوع عنصر ذکر شده باشند. طبیعتاً خود فرم ورود اطلاعات نیز یک کانتینر در نظر گرفته می شود. برای چینش منظم فرمها، فضای داخلی کانتینرها بصورت مجموعه ای از ستونها به تعداد زوج و ردیفها به تعداد نامحدود فرض می شود. سپس هر قلم خصوصیاتی دارد که تعیین میکند چه نسبتی از ستونهای کانتینر پدر و چه تعداد از ردیفهای آن را اشغال می کند. نسبتهای اشغال ستون Normal یا یک ستون، Half یا نصف ستونها و یا Full به معنی تمامی ستونها می باشد.

**کنترلها**: همان فیلدها و روابط موجود در فرم ورود اطلاعات می باشند. توجه شود که همانطور که پیشتر ذکر شد یک رابطه می تواند خود یک فرم ورود اطلاعات است حال یا مستقیم و یا فقط یک میانبر غیر مستقیم.

**فضای خالی**: به منظور ایجاد فاصله ظاهری بین اقلام در صورت نیاز قابل تعریف می باشند.

در ادامه به بررسی دقیقتر خصوصیات هر EntityUICompositionDTO می پردازیم.

**خصوصیت ObjectCategory:**

مشخص کننده نوع هر قلم می باشد که شامل 6 نوع زیر است :

|  |  |
| --- | --- |
| عنوان | توضیحات |
| Group | مشخص کننده کانتینر GroupBox که با یک باکس می تواند. هر گروه می تواند شامل برخی کانتینرها و یا کنترلهای دیگر باشد. |
| TabControl | مشخص کننده کانتینر TabControl که شامل TabPage ها می باشد. |
| TabPage | مشخص کننده کانتینر Tab که می تواند شامل برخی کانتینرها و یا کنترلهای دیگر باشد. |
| Column | مشخص کننده یک کنترل از نوع فیلد ساده |
| Relationship | مشخص کننده یک کنترل از نوع رابطه می باشد |
| EmptySpace | مشخص کننده فضای خالی |

**خصوصیت EntityUISetting:**

تنظیمات ظاهری فرم اصلی در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsCount | تعداد ستونهای داخلی فرم |

**خصوصیت GroupUISetting:**

اگر قلم از نوع Group باشد، تنظیمات ظاهری آن گروه در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsType | میزان ستونهای مورد استفاده توسط GroupBox |
| InternalColumnsCount | تعداد ستونهای داخلی GroupBox |
| UIRowsCount | تعداد ردیفهای مورد استفاده توسط GroupBox |
| Expander | آیا خاصیت کشویی داشته باشد؟ |
| IsExpanded | در صورت کشویی بودن حالت اولیه باز باشد. |

**خصوصیت TabGroupUISetting:**

اگر آیتم از نوع TabControl باشد، تنظیمات ظاهری کنترل در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsType | میزان ستونهای مورد استفاده توسط TabControl |
| UIRowsCount | تعداد ردیفهای مورد استفاده توسط TabControl |
| Expander | آیا خاصیت کشویی داشته باشد؟ |
| IsExpanded | در صورت کشویی بودن حالت اولیه باز باشد. |

**خصوصیت TabPageUISetting:**

اگر آیتم از نوع TabPage باشد، تنظیمات ظاهری در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| InternalColumnsCount | تعداد ستونهای داخلی TabPage |

**خصوصیت EmptySpaceUISetting:**

اگر آیتم از نوع EmptySpace باشد، تنظیمات ظاهری در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsType | میزان ستونهای مورد استفاده توسط فضای خالی |
| ExpandToEnd | آیا فضای خالی شامل کل ستونهای باقی مانده در ردیف شود؟ |

**خصوصیت ColumnUISetting:**

اگر آیتم از نوع Column باشد، تنظیمات ظاهری در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsType | میزان ستونهای مورد استفاده توسط کنترل فیلد معادل |
| UIRowsCount | تعداد ردیفهای مورد استفاده توسط کنترل فیلد معادل |

**خصوصیت RelationshipUISetting:**

اگر آیتم از نوع Relationship باشد، تنظیمات ظاهری در این خصوصیت قرار می گیرد که شامل خصوصیات زیر است.

|  |  |
| --- | --- |
| عنوان | توضیحات |
| UIColumnsType | میزان ستونهای مورد استفاده توسط رابطه |
| UIRowsCount | تعداد ردیفهای مورد استفاده توسط رابطه |
| Expander | آیا خاصیت کشویی داشته باشد؟ |
| IsExpanded | در صورت کشویی بودن حالت اولیه باز باشد. |

لازم به ذکر است که کاربرد اصلی این ساختار در فرمهای تک داده می باشد. در فرمهای چندداده مفهموم کانتینر و سلول بندی ظاهری فرم بصورت توضیح داده شده وجود ندارد و این ساختار درختی به صورت خطی تبدیل شده و کنترلها به ترتیب به عنوان ستونها گرید فرم چند داده ای افزوده می شوند.

تا اینجا به بررسی ساختار خصوصیت UICompositions در موجودیت ورود اطلاعات فرمها پرداختیم. در ادامه به نحوه ساخت و مقداردهی این خصوصیت پرداخته می شود.

#### ساخت و نگهداری الگوی فرمهای ورود اطلاعات

الگوی فرم ورود اطلاعات در متد GetOrCreateEntityUIComposition مقداددهی می شود (BizEntityUIComposition.GetOrCreateEntityUIComposition: fdb08b6bb258). در مرحله اول اگر ساختار الگو ورود اطلاعات برای یک موجودیت که از قبل ساخته و ذخیره شده باشد آنرا دریافت می کند. همچنین اگر ستونها و روابط موجودیت نسبت به الگوی ذخیره شده تغییراتی داشته باشد (ستون یا رابطه ای کم یا زیاد شده باشد) آن موارد را در الگو اعمال می کند. اگر هم الگویی از پیش موجود نباشد بصورت خودکار آنرا می سازد. این متد از سه جای برنامه فراخوانی می شود:

1. هنگام استخراج منابع از پایگاه داده هدف و تنظیم خودکار ظاهری فرم ورود اطلاعات موجودیتهای استخراج شده برای ثبت در فراداده(BizEntitySettings.UpdateDefaultSettingsInModel: 8d0b82740010). در این حالت طبیعتاً ساختار الگو از قبل وجود ندارد و به صورت کامل برای موجودیت ساخته می شود.
2. هنگام تعریف الگوی فرم ورود اطلاعات در فراداده توسط کاربر در فرم مخصوص. به عبارت دیگر آخرین وضعیت الگوی فرم (چه قبلاً تعریف شده باشد و چه نباشد) دریافت می شود و در درخت مربوطه نمایش داده می شود. سپس کاربر می تواند این ساختار را بنابر صلاحدید اصلاح و ذخیره نماید (frmEntityUIComposition.PopulateTree: 35292cf1d9a1).
3. به هنگام دریافت موجودیت فرم ورود اطلاعات فرم که پیشتر توضیحات ارائه شد(BizTableDrivedEntity.GetDataEntryEntity: fedaef41c5c7).

به این متد شناسه موجودیت فرم و ستونها و روابط مورد انتظار برای ساخت الگو فرستاده می شود. به این دلیل لیست ستونها و روابط از خارج به متد ارسال می شوند که، ممکن است فرم ورود اطلاعات برای هر کاربر (بنابر دسترسی ها) دارای ستونها و روابط گزینش شده ای باشند و یا اساساً ستونها و روابط موجودیت دچار تغییراتی شده باشند. بنابراین ساختار الگوی ورود اطلاعات بر اساس آخرین وضعیت ستونها و روابط موجودیت بروزرسانی و دریافت می گردد. در ادامه نحوه ساخت الگوی ورود اطلاعات برای یک موجودیت ورود اطلاعات یک فرم بررسی می گردد.

#### ساخت خودکار الگوی جدید

فرایند ساخت خودکار الگوی فرم ورود اطلاعات دارای پیچیدگیهای فنی و پیاده سازی می باشد بنابراین سعی می شود در تشریح این فرایند عمدتاً به مفاهیم و ایده ها پرداخته شود(BizEntityUIComposition.GenerateUIComposition: ed81338a7008). اولین مرحله ساخت الگو طبیعتاً ایجاد اولین شاخه درخت می باشد که عینیتی از تنظیمات کلی خود فرم ورود اطلاعات است. به عبارت بهتر یک EntityUICompositionDTO از نوع Entity می شود که تعداد ستونهای داخلی فرم مهمترین خصوصیت آن می باشد. در ادامه ستونها و روابط مورد انتظار تحلیل و افزوده می شوند(BizEntityUIComposition.AddColumnsAndRelationships: 52b9e6b626d0). اولین قدم تعیین ترتیب نمایش کنترلهای ستون و روابط است. این ترتیب عمدتاً از خصوصیت Position ستونهای موجودیت تعیین می گردد. از خصوصیت Position می توان برای تعیین موقعیت تمامی ستونهای ساده و همچنین روابط فرعی به اصلی بر مبنای ستونهای کلید خارجی رابطه استفاده نمود. اما شروطی برای نمایش روابط فرعی به اصلی در فرم ورود اطلاعات بر اساس جای ستون مربوطه وجود دارد (BizEntityUIComposition.FKRelationshipHasReservedPosition: 7387bc372bcb):

* اگر رابطه اتحاد به زیر اتحاد بود و تمامی روابط اتحاد به زیر اتحاد هم گروه با رابطه نمایش مستقیم نداشته باشند رابطه قابل جانمایی است. زیرا رابطه مستقیم اتحاد به زیر اتحاد بهتر است در انتهای فرم و در گروهبندی ارائه شود.
* اگر رابطه ارث بری فرزند به پدر بود، چه مستقیم باشد و چه نباشد قابل جانمایی است. زیرا شاکله اصلی موجودیت فرزند موجودیت پدر است.
* اگر رابطه از نوع دیگری باشد تنها اگر مستقیم نباشد قابل جانمایی است.

مرحله بعد افزودن ستونها و روابط جانمایی شده بر اساس خصوصیت Position می باشد(BizEntityUIComposition.FKRelationshipHasReservedPosition: 7387bc372bcb). ستونها در متد AddColumnUICompositionItem افزوده می گردند (BizEntityUIComposition.AddColumnUICompositionItem: 5c93e6562e87). به عبارت دیگر برای هر قلم ستون یک EntityUICompositionDTO از نوع Column ایجاد شده و خصوصیات مربوطه مقداردهی می گردند. نکته مهم اینکه خصوصیت ColumnUISetting برای ستون نیز همینجا ایجاد و مقداردهی می گردد. از جمله خصوصیات آن تعداد ستونهای اشغالی توسط کنترل ستون می باشد که از روی نوع داده ستون و همچنین طول ماکزیمم داده تعیین می گردد. برای مثال برای ستون های رشته کمتر مساوی 64 کاراکتر طول ستون معمولی یا همان تک ستون در نظر گرفته می شود. روابط نیز در متد AddRelationshipUICompositionItemافزوده می گردند (BizEntityUIComposition.AddRelationshipUICompositionItem: 791fcf47bcba). برای هر رابطه نیز خصوصیت RelationshipUISetting مقداردهی می گردد. اگر رابطه مستقیم بود رابطه تمام ستونهای ممکن را اشغال می کند. اگر هم رابطه غیر مستقیم بود نصف ستونهای ممکن برای رابطه رزرو می گردد. پس از جایگذاری ستونها و روابط دارای موقعیت مشخص (Position)، در مرحله بعد سایر روابط گروهبندی می شوند. این روابط شامل روابط فرعی به اصلی جانمایی نشده و همچنین کلیه روابط اصلی به فرعی می شوند. از آنجایی این روابط گروهبندی می شوند که برخی از روابط ممکن است مربوط به دسته های روابط اتحاد و ارث بری باشند و نیاز است در کنار هم و یا بصورت دسته بندی شده نمایش داده بشوند(BizEntityUIComposition.GetOtherRelationshipsGroups: 5d69f05cfdd5). این گروهبندی به این صورت است که روابط اتحاد به زیر اتحاد مربوط به یکدیگر در یک گروه با عنوان رابطه اتحاد مربوطه، روابط ارث بری پدر به فرزند مرتبط در یک گروه با عنوان رابطه ارث بری مربوطه و سایر روابط نیز به صورت تکی در گروه ها جداگانه با عنوان مخصوص هر رابطه قرار میگیرند. پس از گروهبندی، روابط بر اساس گروه های مربوطه طبق قوانین زیر به الگوی فرم افزوده می شوند:

ابتدا به ازای تمامی گروههایی که دارای رابطه مستقیم نمی باشند (همگی روابط گروه غیر مستقیم هستند) تمامی روابط به ترتیب به فرم اضافه می شوند(BizEntityUIComposition.AddOtherRelationshipUICompositionItem: ac47f85a7fd9). برای مثال گروههای روابط اتحاد به زیر اتحاد که حتی فقط یک رابطه مستقیم باشد در نظر بگیرید. این روابط با وجود داشتن Position احتمالی بابت ستون های کلید فرعی طبق توضیحات مرحله قبل جایگذاری نشدند. در اینجا هم که همه روابط پردازش می شوند این روابط دارای اولویت نمی باشند زیرا همگی روابط گروه غیر مستقیم نیستند. در مقابل در مثالی دیگر تمامی گروه های تک رابطه که غیر مستقیم می باشند در این مرحله افزوده می شوند. در ادامه گروه هایی باقی می مانند که حداقل یک رابطه مستقیم دارند و پردازش آنها انجام می گردد.

* اگر تعداد گروه هایی که دارای رابطه غیرمستقیم می باشند بیش از یک مورد باشد، یک کنترل تب با عنوان "روابط" به فرم افزوده می شود. سپس به ازای هر گروه یک برگه جدید به نام گروه به تب کنترل روابط افزوده می گردد (به آن برگه گروه اطلاق می کنیم). در ادامه :
  + اگر گروه دارای بیش از یک رابطه باشد (ارث بری یا اتحاد باشد)، تب کنترل جدیدی برای گروه در برگه گروه ایجاد می شود. سپس به ازای تمامی رابطه های زیر گروه یک برگه به تب کنترل گروه افزوده شده و رابطه به این برگه جدید افزوده می گردد. به عبارت دیگر برای گروه های چند رابطه ای یک تب کنترل مخصوص آن گروه در برگه گروه ایجاد می گردد.
  + اگر گروه دارای یک رابطه باشد به ازای آن رابطه یک برگه به تب کنترل روابط افزوده شده و رابطه به این برگه افزوده می گردد. به عبارت دیگر برای گروه های تک رابطه ای تنها یک برگه مخصوص به تب کنترل روابط افزوده می گردد.
* اگر تعداد گروه هایی که دارای رابطه غیرمستقیم می باشند تنها یک مورد باشد، یک تب کنترل به نام آن گروه اضافه می گردد. سپس به ازای تمامی رابطه های زیر گروه یک برگه به تب کنترل گروه افزوده شده و رابطه به این برگه جدید افزوده می گردد.

#### ساخت دستی الگوی فرم ورود اطلاعات

الگوهای خودکار ایجاد شده برای موجودیتها را می توان بصورت دستی نیز اصلاح و بروزرسانی نمود. برای این کار فرم مخصوصی طراحی شده است(frmEntityUIComposition: 3292f851d67b). در این فرم الگوی موجود در درختی نمایش داده شده و کاربر می تواند اقلام مورد نظر را از انواع مختلف خود چینش کند. سپس این الگوی تعریف شده قابلیت ذخیره سازی دارد(BizEntityUIComposition.UpdateUIComposition: f794d23c8a6a).

#### بروزرسانی الگوی موجود به هنگام دریافت

همانطور که پیش تر ذکر شد، اتفاق دیگری که در متد GetOrCreateEntityUIComposition می افتد بروزرسانی الگوی از پیش تعریف شده بر اساس آخرین وضعیت ستونها و روابط موجودیت می باشد. در درجه اول به هنگام دریافت الگو، تنها ستونها و روابطی تبدیل می گردند که در لیست روابط و ستونهای موجودیت باشند. به عبارت دیگر ستونها و روابطی که دیگر وجود ندارند از الگو حذف می گردند (BizEntityUIComposition.ToEntityUICompositionDTO: 6204e634719f). سپس تمام ستونها و روابطی در موجودیت که در الگو موجود نمی باشند به الگو افزوده می گردد. برای فزودن از همان روش پیشتر توضیح داده شده برای افزودن ستونها و روابط به الگو استفاده می گردد (BizEntityUIComposition.AddColumnsAndRelationships: 52b9e6b626d0). نکته مهم اینست که روابط و ستونهای جدید به انتهای زیر مجموعه های الگوی موجودیت افزوده می شوند و به هیچ عنوان دخل و تصرفی در الگوی ثبت شده موجود انجام نمی پذیرد.

الگوی فرم ورود اطلاعات آخرین بخش از تشریح موجودیت فرم ورود اطلاعات بود. حال در ادامه کاربرد موجودیت ورود اطلاعات را برای یک فرم بررسی می نماییم.

### ساخت کنترلها برای فرم ورود اطلاعات مستقیم :

پس از ساخت دکمه های فرم ورود اطلاعات مستقیم نوبت به ساخت کنترلهای ورود اطلاعات فرم می رسد(BaseEditEntityArea.GenerateControls: ddfdfcbe4f89). همانطور که پیش تر اشاره شد لازمه چینش این کنترل ها و جزئیات آنها در موجودیت ورود اطلاعات فرم نهفته است. تمامی جزئیات لازم برای چینش فرم در خصوصیت UICompositions موجودیت ورود اطلاعات قرار دارد. برای عملیات چینش از متد GenerateUIControlsByCompositionDTO استفاده می گردد. این متد برای فرم های تک داده و چند داده بصورت مجزا پیاده سازی شده است زیرا اساساً ماهیت چینش کنترلها در این دو نوع فرم متفاوت می باشد. برای فرمهای چند داده به علت عدم وجود کانتینرها و فضای خالی این چینش خطی و ساده تر است(EditEntityAreaMultipleData.GenerateUIControlsByCompositionDTO: 3c156446abf7). برای فرمهای تک داده نیز عیناً ساختار الگو در فرم نمایش داده می گردد(EditEntityAreaOneData.GenerateUIControlsByCompositionDTO: ad8b0609f380). دو مطلب قابل توجه در این فرایند نحوه تولید کنترلهای ستونی و روابط می باشند که در ادامه توضیحاتی درباره آنها ارائه خواهد شد.

#### ایجاد کنترلهای ستون :

در پردازش الگوی فرم زمانی که به قلمی از نوع Column می رسیم یک کنترل ورود اطلاعات ساده (مانند تکست باکس یا تاریخ) ایجاد می گردد که به خصوصیتی از داده های موجودیت متصل شود. طبیعتاً کنترلهای ستون نیز برای دو نوع فرم تک داده و چند داده متفاوت می باشند. برای فرم تک داده شیء از نوع SimpleColumnControlOne تولید می گردد که عیناً معادل یک کنترل ورودی واسط کاربر مانند یک تکست باکس است(SimpleColumnControlOne: 0c72888906d5). برای فرم های چند داده نیز این شیء از نوع SimpleColumnControlMultiple می باشد که عیناً یک ستون گرید فرم ورود اطلاعات چند داده ای می باشد(SimpleColumnControlMultiple: 0616c9171488). جزئیات بیشتر این کلاسها و نحوه عملکردشان در مدیریت کنترلهای واسط کاربری در ارتباط خصوصیات داده و واسط کاربری در بخش های مربوطه بعدی تشریح خواهد شد.

#### ایجاد کنترلهای روابط:

کلید مهم در درک مفاهیم این بخش اینست که هر رابطه در فرم ورود اطلاعات، خود یک فرم ورود اطلاعات فرض می شود. در پردازش الگوی فرم زمانی که به قلمی از نوع Relationship می رسیم یک کنترل نمایش فرمی دیگر در داخل فرم جاری ایجاد می گردد. طبیعتاً کنترلهای مذکور برای دو نوع فرم تک داده و چند داده متفاوت می باشند. برای فرم تک داده نمای اولیه فرم که یا فرم ورود اطلاعات مستقیم است یا غیر مستقیم نمایش داده می شود و برای فرم چند داده قطعاً نمای غیر مستقیم فرم نمایش داده می شود. جزئیات بیشتر ایجاد این فرمهای رابطه در ادامه آمده است.

چه در فرم تک داده و چه چند داده، در پردازش الگو فرم هرگاه به یک رابطه می رسیم اولین گام تولید فرم طرف دیگر رابطه است. این ایجاد فرم تفاوت آنچنانی با مراحل ایجاد یک فرم ورود اطلاعات که در بخشهای پیشین و همین بخش نیز توضیح داده شده است ندارد. گویی یکبار دیگر مراحل ایجاد و نمایش یک فرم ورود اطلاعات تکرار می شود. تنها تفاوت اینست که خصوصیات EditEntityAreaInitializer اینبار از رابطه جاری تعیین می گردند(BaseEditEntityArea.GenerateRelationshipControlEditArea: 7b5673de233c). در ادامه برای فرم های میزبان تک داده (فرم طرف اول رابطه) شیء از نوع RelationshipColumnControlOne تولید می گردد که میزبان فرم رابطه تولید شده (فرم طرف دوم رابطه) می باشد(RelationshipColumnControlOne: 7a0bc2462763). برای فرم چند داده نیز شیء از نوع RelationshipColumnControlMultiple تولید می گردد(RelationshipColumnControlMultiple: c0b7ea24c7e9). هر کدام از این انواع کنترل رابطه دارای جزئیات و عملکرد تخصصی می باشند که در بخش های بعد به آنها بیشتر پرداخته خواهد شد.

تا به اینجا عمده اقدامات تولید فرم مستقیم ورود اطلاعات که شامل تولید دکمه ها و کنترل های ورود اطلاعات می شوند معرفی گردیدند. البته پس از این مراحل رویدادهای دیگری نیز مانند تعیین دسترسی دکمه ها انجام می گیرد اما چون این رویدادها مستقلاً به این بخش مربوط نمی شوند در بخش های دیگر به آن پرداخته خواهد شد.

در ادامه متد SetAreaInitializerAndGenerateView، به حالتی می پردازیم که نیاز ما ایجاد ظاهر اولیه فرم غیر مستقیم ورود اطلاعات است.

## **ساخت ظاهر فرم غیر مستقیم (GenerateTempView):**

اگر InteractionMode اولیه فرم یکی از حالات CreateInDirect، CreateSelectInDirect یا Select باشد ظاهر اولیه که از فرم ساخته می شود نمای غیر مستقیم می باشد. برای این امر متد GenerateTempView فراخوانی می شود (BaseEditEntityArea.GenerateTempView: d932102d6454). این نما دارای دکمه های میانبر و یک کنترل تسکت باکس می باشد که برخی از آنها بر حسب IntracionMode فرم، فعال یا غیر فعال می گردد. در زیر لیست تمامی دکمه ها و توضیحات مربوطه آمده است:

|  |  |  |
| --- | --- | --- |
| نام | عنوان | توضیحات |
| ButtonInfo | اطلاعات |  |
| ButtonClear | پاک کردن |  |
| ButtonSearchForm | نمایش فرم جستجو | در صورتی که داده ها قابل جستجو و انتخاب باشد (Select یا CreateSelectInDirect) فعال است. فرم جستجوی داده ها را نمایش می دهد. |
| ButtonPopup | پنجره نتیجه جستجو | در صورتی که داده ها قابل جستجو و انتخاب باشد (Select یا CreateSelectInDirect) فعال است. پنجره نتیجه جستجو را باز و بسته می کند. |
| ButtonDataEdit | فرم اصلاح داده | در صورتی که داده ها قابل اصلاح باشند فعال است (CreateInDirect یا CreateSelectInDirect). فرم اصلاح داده را نمایش می دهد. |
| SearchTextbox | تکست باکس جستجو |  |

## دیگر شرایط ساخت ظاهر مستقیم و غیر مستقیم :

اشاره شد که در متد SetAreaInitializerAndGenerateView تنها ظاهر اولیۀ مستقیم و یا غیر مستقیم فرمها ساخته می شوند. هنگامی که InteractionMode فرم مبنی بر ساخت ظاهر اولیه مستقیم باشد دیگر ظاهر غیر مستقیم فرم معنایی ندارد امّا زمانی که ظاهر غیرمستقیم نمای اولیه فرم است ممکن است در صورت احراز شرایط و تصمیم کاربر نیاز به اصلاح اطلاعات و بنابراین ایجاد ظاهر مستقیم فرم ورود اطلاعات نیز باشد. بنابراین رویداد ساخت ظاهر مستقیم یا همان GenerateDataView ممکن است بنابر شرایط اشاره شده و پس از زدن دکمه میانبر نمایش فرم ورود اطلاعات در فرم غیر مستقیم، مستقلاً فراخوانی گردد(ChildRelationshipInfo.DataViewRequested: e48af9744b57). البته تولید فرم ورود اطلاعات مذکور تنها یکبار و پس از اولین فراخوانی انجام می گیرد.

بخش مهمی از عملکرد فرم های ورود اطلاعات جستجو، نمایش و انتخاب داده ها به منظور اصلاح اطلاعات می باشد. بنابراین زیر فرمهای جستجو و نمایش داده جزء لاینفکی از فرمهای ورود اطلاعات می باشند. اینکه فیلترهای جستجو چگونه تعریف و دسته بندی شوند و یا داده ها به چه کیفیتی نمایش داده شوند از پیچیدگی های این زیر فرمها می باشند. در بخش بعدی به نحوه عملکرد این زیر فرمها پرداخته می شود.

## زیر فرم جستجو و انتخاب داده :

یکی از عملکردهای مهم فرم کار با موجودیت و داده ها، جستجو و انتخاب داده های موجود می باشد، حال چه فرم مستقل باشد و یا از رابطه بوجود آمده باشد. در واقع اگر InteractionMode اولیه فرم یکی از حالات CreateSelectDirect ، CreateSelectInDirect یا Select باشد، داده قابل جستجو است و در نتیجه زیر فرم جستجو و نمایش داده معنی پیدا می کند. زیر فرم جستجو داده خود از دو بخش تشکیل شده است. بخش جستجوی داده ها (SearchEntityArea) و بخش نمایش داده های جستجو شده به منظور انتخاب داده(ViewEntityArea). ظاهر این زیر فرمها همراه با تولید فرم ورود اطلاعات بصورت اولیه ایجاد نمی شوند، بلکه زمانی که برای اولین بار نمای آنها فراخوانی می گردند فرایند ساخت آنها نیز انجام می گردد. در ادامه به هر کدام از این دو بخش جداگانه پرداخته می شود.

### فرم جستجوی داده (SearchEntityArea):

فرم جستجوی داده محیطی را برای ورود فیلترهای جستجو و نهایتاً انجام عملیات جستجو فراهم می کند. در فرم جستجو قرار است پارامترهایی در واسط کاربری نمایش داده شوند تا کاربر بتواند به جستجوی داده ها بپردازد. لیست و جزئیات این پارامتر ها یا از قبل در مدل فراداده برای موجودیت تعریف شده اند و یا در لحظه توسط کاربر قابل اعمال می باشند. بنابراین این فرم دارای دو بخش جستجو می باشد، جستجوی تعریف شده و جستجوی پیشرفته. کلاس پایه فرم جستجوی در اولین ارجاع به آن ساخته می گردد(BaseEditEntityArea.SearchEntityArea: 54e3b4a36ac9). امّا ظاهر آن که شامل ظاهر دو زیر فرم جستجوی تعریف شده و پیشرفته است زمانی ساخته می شود که نیاز به نمایش فرم باشد(SearchEntityArea.SearchView: 3100cfe8e22f).

برای کار با فرم جستجو ابتدا نیاز به ایجاد یک SearchEntityAreaInitializer می باشد.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| EntityID | int | شناسه موجودیت |
| Title | string | عنوان فرم |
| SearchEntityID | int | شناسه تنظیمات جستجوی پیش فرض |
| PreDefinedSearchMessage | PreDefinedSearchDTO | مقادیر پیش فرض جستجوی تعریف شده |
| AdvancedSearchDTOMessage | AdvancedSearchDTO | مقادیر پیش فرض جستجوی پیشرفته |
| ForSave | bool | آیا مقادیر پیش فرض برای ذخیره سازی می باشند |

ساده ترین حالت ساخت این فرم مقداردهی خصوصیت EntityID می باشد که تنها با اشاره به شناسه موجودیت فرم جستجوی پیش فرض موجودیت ساخته می شود. سایر خصوصیات دیگر مانند PreDefinedSearchMessage و AdvancedSearchDTOMessage در کاربردهای پیچیده تر جستجو استفاده می شوند. توجه شود که فرم جستجوی داده تنها مختص فرم های ورود اطلاعات نمی باشد و ممکن است در بخشهای دیگر به مانند گزارشات نیز بکار بیاید. بصورت کلی در این بخش تنها به بررسی فرم جستجوی داده در قالب فرمهای ورود اطلاعات می پردازیم که در این حالت تنها خصوصیت EntityID برای شروع فرم جستجو مقداردهی می شود. در ادامه به بررسی عملکرد دو زیر فرم جستجوی تعریف شده و پیشرفته می پردازیم.

### زیر فرم جستجوی تعریف شده :

منظور از جستجوی تعریف شده، جستجویی است که فیلترهای جستجو (شامل فیلدهای ساده موجودیت و فیلدهای روابط مرتبط با موجودیت)، از پیش تعریف شده اند. این جستجو فیلترهای از پیش تعیین شده ای را به صورت اختصاری و معنی دار در اختیار کاربر می گذارد(EntityDefinedSearchArea: de46ee61a50f). اساس این فرم شامل کنترلهایی به منظور ورود اطلاعات جستجو و همچنین دکمه هایی برای انجام عملکردها می باشد.

#### دریافت الگوی فرم جستجوی تعریف شده :

برای ایجاد کنترلها یا همان پارامترهای جستجو نیاز به الگویی برای تعریف و جاگذاری آنها می باشد (به مانند فرم ورود اطلاعات). این الگو در موجودیت EntitySearchDTO فرم قرار دارد. این الگو یا از قبل توسط کاربر در برنامه مدیریت فراداده ساخته می شود و یا در لحظه بصورت خودکار تولید می گردد. برای هر موجودیت می توان بیش از یک الگو تعریف کرد امّا در فرم ورود اطلاعات برای جستجوی داده های موجودیت تنها از یک الگو می توان استفاده نمود که همان الگوی پیش فرض موجودیت است. البته در حالتی که فرم جستجو در پارامترهای خود PreDefinedSearchMessage را مقداردهی کرده باشد بصورت ضمنی الگوی جستجو نیز مشخص شده و فرم بر اساس آن الگو طراحی می گردد. در اینجا با این حالت کاربرد فرم جستجو کاری نداریم و همانطور که پیشتر اشاره شد فرض میکنیم تنها شناسه موجودیت مشخص است، بنابراین الگوی فرم جستجو تعریف شده همان الگوی پیش فرض موجودیت است که یا توسط کاربر قبلاً تعریف شده و یاد در صورت عدم وجود بصورت خودکار تولید می گردد (EntityDefinedSearchArea.EntitySearchDTO: 5f09df3218c6).

**ساختار EntitySearchDTO:**

ساختار این موجودیت در زیر نمایش داده شده است.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه الگو |
| TableDrivedEntityID | int | شناسه موجودیت مرتبط |
| Title | string | عنوان الگو |
| EntitySearchAllColumns | List<EntitySearchColumnsDTO> itory | لیست ستونها |
| IsDefault | bool | آیا الگوی پیش فرض است؟ |
| EntityUISetting | EntityUISettingDTO | تنظیمات ظاهری فرم الگو |

مبنای اصلی این موجودیت لیستی از EntitySearchColumnsDTO ها یا همان فیلتر های جستجو می باشد. در زیر ساختار هر فیلتر جستجو آمده است.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه فیلتر |
| ColumnID | int | شناسه ستون هدف فیلتر |
| Column | ColumnDTO | موجودیت ستون هدف |
| OrderID | short | ترتیب |
| Alias | string | عنوان فیلتر |
| RelationshipPath | string | مسیر رشته رابطه |
| RelationshipTail | EntityRelationshipTailDTO | موجودیت رشته رابطه |
| Tooltip | string | توضیحات کمکی |
| ExcludeInQuickSearch | bool | استثناء در جستجوی سریع |
| ColumnUISetting | ColumnUISettingDTO | تنظیمات ظاهری ستون هدف |
| RelationshipUISetting | RelationshipUISettingDTO | تنظیمات ظاهری رابطه |

برای دریافت الگوی پیش فرض در فرمهای ورود اطلاعات از متد GetOrCreateEntitySearchDTO استفاده می شود که در آن یا الگوی پیش فرض از قبل تعریف شده که همان برگردانده می شود و یا بصورت خودکار ساخته می شود(BizEntitySearch.GetOrCreateEntitySearchDTO: ec6a26985ad9). کاربرد دیگر این متد در زمان استخراج موجودیتها در برنامه مدیریت فراداده می باشد که بصورت خودکار یک الگوی جستجو برای هر موجودیت ساخته می شود (BizEntitySettings.UpdateDefaultSettingsInModel: 8d0b82740010). برای ساخت خودکار فیلترهای الگو از متد GenereateDefaultSearchColumns استفاده می شود (BizEntitySearch.GenereateDefaultSearchColumns: 9e5b4c36a21c). بطور کلی دو نوع فیلتر جستجو قابل تعریف است. نوع اول جستجوی مستقیم ستون های موجودیت و یا موجودیت های مرتبط است. در این حالت لیست ستونها و روابط موجودیت دیگر نیز به مانند موجودیت اصلی پیمایش می شوند تا ستونهای کاربردی آن که به جستجوی موجودیت اصلی کمک می کند نیز به لیست فیلتر های جستجو اضافه شوند. برای مثال فرض کنید در جستجوی اشخاص، موجودیت رابطه شهر تولد شخص نیز بررسی می شود و ستونهای کد و نام شهر تولد شخص هم به لیست فیلترهای جستجوی شخص اضافه می گردد. در این حالت خصوصیت ColumnID فیلتر قطعاً مقدار دارد و به ستون هدف جستجو اشاره میکند. نوع دیگر فیلتر جستجو تعیین یک رابطه مرتبط با موجودیت است به منظور تعیین دقیق یک قلم داده مرتبط. برای مثال فرض کنید که برای جستجوی شخص دقیقاً شهر تولد انتخاب شود و به ذکر کامل یا بخشی از عنوان یا کد شهر اکتفا نشود. در این حالت ستون ColumnID مقدار صفر دارد و صرفاً رشته رابطه مشخص است. البته در نهایت جستجوی قلم داده مرتبط بصورت ضمنی همان جستجوی ستون کلید خارجی رابطه است که به یک قلم داده یکتای مرتبط منتهی می شود. قانون ساخت ترکیب جستجوی پیش فرض در زیر آمده است:

اگر موجودیت نما باشد، تمامی ستونها به عنوان فیلتر جستجو انتخاب می شوند. در غیر اینصورت به ازای هر ستون قوانین زیر کنترل می شوند:

* اگر فیلد کلید اصلی موجودیت باشد به جستجو اضافه می شود.
* اگر فلید کلید خارجی باشد و مربوط به روابط ارث بری فرزند به پدر، چند به یک یا یک به یک صریح باشد
  + ستون به لیست جستجو اضافه می شود.
  + اگر مربوط به رابطه ارث بری باشد، موجودیت طرف رابطه هم برای پیمایش ستونها مجدداً به همین رویداد فرستاده می شود تا این قوانین برای روابط و ستونها آن موجودیت نیز بررسی و در صورت شامل شدن در قوانین به لیست فیلدهای جستجوی موجودیت اولیه اضافه شوند.
  + اگر مربوط به رابطه چند به یک یا یک به یک صریح باشد، اگر در لیست سابقه روابط کمتر از 2 رابطه غیر ارث بری باشد (تنها یک شاخه از همین روابط چند به یک یا یک به یک موجود باشد)، خود موجودیت رابطه به لیست جستجو اضافه می شود (جستجوی قلم داده مرتبط) و همچنین موجودیت طرف رابطه هم مجدداً به همین رویداد فرستاده می شود تا این قوانین برای ستونها و روابط آن موجودیت نیز بررسی و اعمال گردد.
* اگر نام یا عنوان ستون شامل قوانین نام ستونهای جستجو شود به لیست جستجو اضافه می شود(BizEntitySearch.CheckColumnDetection: 672a797215e4).
* طبیعتاً اگر ستون در بیش از یک قانون شامل شود تنها یکبار به لیست جستجو اضافه می گردد. همچنین اگر ستونی افزوده شود و کلید رابطه باشد (چه کاید اصلی چه خارجی) در بررسی موجودیت طرف دیگر، ستون کلید طرف دیگر افزوده نخواهد شد.

به منظور درک بهتر قوانین ارائه شده در بحث بررسی موجودیت های طرف رابطه اینگونه می توان بیان کرد که زمانی که یک ستونهای یک موجودیت پیمایش می شوند موجودیت های رابطه پدر نیز به مانند موجودیت فرزند پیمایش می شوند (زیرا بخشی از موجودیت جاری فرض می گردند) . همچنین در طول هر رشته رابطه مورد بررسی، موجودیتهای روابط چند به یک و یک به یک (صریح) نیز تنها برای یکبار پیمایش می شوند.

#### ساخت الگوی فرم جستجوی تعریف شده بصورت دستی:

مراحل قبلی توضیحاتی درباره ساخت خودکار الگوی جستجو بود. الگوی جستجو را می توان به صورت دستی نیز در برنامه مدیریت فراداده تعریف نمود. این اتفاق در فرمی مخصوص انجام می گردد(frmEntitySearch: 62576df5955b) و سپس الگوی جدید یا موجود در مدل بروزرسانی می گردد(BizEntitySearch.UpdateEntitySearchs: 504fdd7bcfcc).

#### ساخت کنترلهای فرم جستجوی تعریف شده:

برای ساخت جزئیات فرم جستجوی تعریف شده از متد GenerateUIControls استفاده می شود(EntityDefinedSearchArea.GenerateUIControls: 31d456f709e0). در ساخت فیلتر های جستجو از همان الگوی جستجوی دریافت شده استفاده می شود. ابتدا تکست باکس جستجوی سریع ساخته می شود. این تسکت باکس به منظور تسهیل جستجوی داده ها توسط کاربر ایجاد می گردد. عملکرد آن اینست که در صورت مقداردهی آن تمامی ستونهای الگو که از جستجوی سریع مستثنی نشده باشند با شرط "یا" جستجو می شوند. در مرحله بعد تمامی کنترلهای جستجو شامل فیلدهای ساده و یا روابط ساخته می شوند. برای هر ستون جستجو یعنی فیلترهایی که ColumnID آنها مقدار دارد یک SimpleSearchColumnControl ساخته می شود که یک کنترل ورودی را در واسط کاربری نمایندگی می کند. برای هر فیلتر قلم داده نیز یک RelationshipSearchColumnControl ایجاد می گردد که فرمی را به منظور انتخاب داده از موجودیت هدف فیلتر را نمایش خواهد داد.

#### دکمه های فرم جستجوی تعریف شده :

دکمه های فرم جستجوی تعریف شده دو مورد می باشند (EntityDefinedSearchArea.AddCommands: 101c0aa583fe). دکمه اول بنام SearchClearCommand که مسئول پاک کردن مقادیر فیلترها می باشد (SearchClearCommand: be7702379276) و دیگری SimpleSearchConfirmCommand که عملیات جستجو را انجام می دهد (SimpleSearchConfirmCommand: 72429c533aa4). به عبارت دیگر با کلیک بر روی این دکمه ابتدا عبارت جستجو داده ها ساخته می شود و سپس رویداد جستجوی فرم فراخوانی می گردد. برای ساخت عبارت جستجو از متد GetSearchRepository استفاده می شود (EntityDefinedSearchArea.GetSearchRepository: b1b5fa67fe2d) که بخشی از عبارت جستجو از مقدار وارد شده در جستجوی سریع ساخته می شود (AgentHelper.GetQuickSearchLogicPhrase: 8384c544000f) و بخش دیگر آن از کنترلهای جستجوی ستونی و قلم داده ای فرم.

#### سایر کاربردهای جستجوی تعریف شده:

به غیر از فرم جستجو در فرمهای ورود اطلاعات، کاربرد دیگر آن در قالب فرم جستجو در فرم تعریف جستجوی پیش فرض در برنامه مدیریت می باشد(frmSearchRepository: 6ee975bba1ca). هدف این فرم تعریف مقادیر پیش فرضی برای یک جستجوی تعریف شده و یا یک جستجوی پیش رفته می باشد. در این حالت به هنگام ایجاد زیر فرم جستجوی تعریف شده، خصوصیت ForSave در SearchEntityAreaInitializer مقدار True دارد و به این ترتیب مشخص است که قصد تعریف مقادیر و یا فرمول های خامی را به عنوان عبارت پیش فرض جستجو داریم. با تایید کاربر عبارت جستجو به همراه مقادیر پیش فرض (که از جنس فرمول نیز می توانند باشند) از فرم جستجوی تعریف شده دریافت شده (EntityDefinedSearchArea.GetSearchRepositoryForSave: ab359aa2461b) و در مدل فراداده ذخیره می گردد (BizSearchRepository.UpdatePreDefinedSearch: de19b52466fc). نمایش مقادیر ذخیره شده نیز از طریق متد ShowPreDefinedSearch صورت می گیرد(EntityDefinedSearchArea.ShowPreDefinedSearch: cd004ddc85ff). از این متد در دوحالت استفاده می گردد، یا در فرم تعریف مقادیر جستجوی تعریف شده در برنامه مدیریت فراداده (به منظور نمایش و اصلاح موارد قبلی) استفاده می گردد که ترکیب مستقیماً به متد نمایش ارسال می شود و یا در برنامه ورود اطلاعات در فرمهای جستجویی که مقادیر پیش فرض معنی دارد به مانند فرم جستجو در گزارشات. در این حالت ترکیب ذخیره شده از طریق پارامتر PreDefinedSearchDTO موجود در SearchEntityAreaInitializer (که پیشتر اشاره شده) به منظور نمایش مقادیر پیش فرض در اختیار فرم جستجوی تعریف شده قرار می گیرد..

### زیر فرم جستجوی پیشرفته :

بخش دیگری از فرم جستجوی داده زیر فرم جستجوی پیشرفته یا همان AdvancedSearch است (AdvancedSearchEntityArea: ef3528f2b0f5). منظور از جستجوی پیشرفته ساخت عبارات جستجو بصورت تودرتو و ترکیبی با حداقل محدودیتها توسط خود کاربر می باشد. یک عبارت جستجوی هر چند پیچیده را می توان بصورت درختی تو در تو از ترکیب تک عبارتها با شرطهای “and” و یا “or” ساده سازی نمود. بنابراین در زیر فرم جستجوی پیشرفته نیز برای نمایش و یا ساخت عبارت جستجو از یک درخت استفاده می شود. نودهای این درخت هر کدام معادل یک عبارت یا Phrase می باشند (AdvancedSearchEntityArea.AddNode: dc5ef44a6503). حال این Phrase خود می تواند یک تک عبارت بروی یک ستون (و مقدار مورد جستجو) باشد و یا یک جمله عبارت به معنای ترکیبی از عبارت ها با تعیین شرط And یا Or که مجدداً هر عبارت خود می تواند یک تک عبارت باشد و یا جمله عبارت. همچنین یک جمله عبارت ممکن است ضمن حفظ خصوصیات ذاتی خود به دو موجودیت دیگر اختصاصی شود. یک همان جمله عبارت موجودیت اصلی باشد که صرفاً به جهت تعیین شاخه اصلی اختصاصی می گردد و شناسه موجودیت اصلی را در خود دارد و دیگری تعریف یک جمله عبارت بروی یک موجودیت مرتبط با موجودیت جاری باشد. به این ترتیب می توان موجودیت جاری را نیز بر اساس ترکیب خصوصیات موجودیتهای مرتبط نیز فیلتر نمود.

جدول زیر انواع Phrase اختصاصی شده را معرفی می کند.

|  |  |  |
| --- | --- | --- |
| نوع | نوع پدر | توضیحات |
| Phrase |  | نوع کلی عبارت که تنها از آن ارث بری می شود |
| LogicPhraseDTO | Phrase | جمله عبارت منطقی که خود یک Phrase اما می تواند شامل چندین Phrase با نوع رابطه And و یا OR باشد. |
| SearchProperty | Phrase | عبارت روی یک تک خصوصیت |
| DP\_SearchRepositoryRelationship | LogicPhraseDTO | عبارت ارتباط یک موجودیت با موجودیت دیگر، خود یک جمله عبارت منطقی است و فقط شناسه رابطه را افزون در خود دارد |
| DP\_SearchRepositoryMain | LogicPhraseDTO | عبارت اصلی که جستجو از آن شروع می شود. همان عبارت منطقی اولیه است و از نوع جمله عبارت است و فقط شناسه موجودیت جستجو در آن مشخص است. |

بنابراین طبق تعریف بالا هر عبارت جستجو در شاخه اصلی از نوع SearchRepositoryMain می باشد که می تواند شامل زیر عبارت های دیگر یا همان Phrase دیگر از نوع SearchProperty، LogicPhraseDTO و یا DP\_SearchRepositoryRelationship باشد. هر کدام از انواع LogicPhraseDTO یا SearchRepositoryRelationship نیز چون از نوع جمله عبارت منطقی هستند می توانند مجدد شامل عبارات دیگر باشند. نوع DP\_SearchRepositoryRelationship مشخص می کند که قصد داریم یک جمله عبارت بر روی یک موجودیت مرتبط با موجودیت جاری معرفی نماییم. به همین ترتیب عبارت جستجوی تودرتو ساخته می شود.

#### نحوه کار با درخت جستجو:

برای افزودن و یا حذف نودهای درخت، می توان با راست کلیک بروی هر نود گزینه هایی از منو های نمایش داده شده را انتخاب نمود. این منوها بسته به نوع شاخه یا همان عبارت مرتبط با شاخه که تک خصوصیت است (SearchProperty) و یا جمله عبارت(LogicPhraseDTO) متفاوت خواهند بود. در واقع تک عبارت تنها دارای منوی حذف می باشد امّا جمله عبارت علاوه بر حذف دارای منوهای اختصاصی متفاوتی می باشد. از طریق این منوها امکان تعریف انواع عبارتهای تشریح داده شده بصورت تودرتو فراهم می گردد. در ادامه این منوها معرفی می گردند امّا پیش از آن لازم است درباره مفهوم "موجودیت جاری" توضیح داده شود. منظور از موجودیت جاری آخرین موجودیتی در شاخه های پدر است که زیر شاخه ها یا همان عبارتهای جستجو بر روی آن تعریف می شود. در ابتدا موجودیت جاری همان موجودیت اصلی جستجو است. با افزودن هر عبارت جستجوی رابطه موجودیت جاری برای آن شاخه و تمام زیر شاخه هایش به موجودیت طرف دیگر رابطه تغییر می کند.

#### منوهای مهم جمله عبارت :

* **افزودن جمله عبارت :** با انتخاب این منو یک جمله عبارت از نوع ساده LogicPhraseDTO زیر شاخه انتخاب شده ایجاد می گردد. به این ترتیب می توان بصورت تودرتو جمله عبارتها را بروی موجودیت جاری ایجاد نمود.
* **افزودن لیست خصوصیات :** زیر فرم اختصاصی RawSearchEntityArea نمایش داده می شود. این زیر فرم به مانند یک فرم جستجوی ساده برای موجودیت جاری متشکل از تمامی ستونهای آن موجودیت می باشد (RawSearchEntityArea.GenerateUIControls: 527171b12fdc). خروجی آن نیز لیستی از SearchProperty ها می باشد (RawSearchEntityArea.GetSearchRepository: 79a3cc81a307) که با زدن دکمه تایید دریافت و به فرم جستجوی پیشرفته ارسال می گردد (RawSearchConfirmCommand: e180dd1853e5).
* **افزودن جمله عبارت رابطه :** به ازای هر رابطه برای موجودیت جاری می توان یک جمله عبارت رابطه یا همان SearchRepositoryRelationship اضافه نمود. به این ترتیب موجویت جاری به موجودیت طرف دیگر رابطه تغییر می کند. با این روش می توان داده های موجودیت جاری را با استفاده از خصوصیات موجودیت های مرتبط با آن فیلتر نمود(AdvancedSearchEntityArea.RelationshipSearchRepository\_AddNew: bf90baf6917a).

#### دکمه های فرم جستجوی پیشرفته :

دکمه های فرم جستجوی تعریف شده دو مورد می باشند (AdvancedSearchEntityArea.AddCommands: 24fb53a7c80e). دکمه اول بنام SearchClearCommand که مسئول پاک کردن مقادیر فیلترها می باشد (SearchClearCommand: be7702379276) و دیگری AdvancedSearchConfirmCommand که عملیات جستجو را انجام می دهد (AdvancedSearchConfirmCommand: 263206ad4af8). به عبارت دیگر با کلیک بر روی این دکمه ابتدا عبارت جستجو داده ها ساخته می شود و سپس رویداد جستجوی فرم فراخوانی می گردد. برای ساخت عبارت جستجو از متد GetSearchRepository استفاده می شود (AdvancedSearchEntityArea.GetSearchRepository: 51644c6aa16f) که تنها کافی است عبارت معادل شاخه اصلی که از نوع DP\_SearchRepositoryMain می باشد برگردانده شود.

#### سایر کاربردهای جستجوی پیشرفته:

به غیر از فرم جستجو در فرمهای ورود اطلاعات، کاربرد دیگر آن در قالب فرم جستجو در فرم تعریف جستجوی پیش فرض در برنامه مدیریت می باشد(frmSearchRepository: 6ee975bba1ca). هدف این فرم تعریف مقادیر پیش فرضی برای یک جستجوی تعریف شده و یا یک جستجوی پیش رفته می باشد. در این حالت به هنگام ایجاد زیر فرم جستجوی پیشرفته، خصوصیت ForSave در SearchEntityAreaInitializer مقدار True دارد و به این ترتیب مشخص است که قصد تعریف مقادیر و یا فرمول های خامی را به عنوان عبارت پیش فرض جستجو داریم. با تایید کاربر، عبارت جستجو به همراه مقادیر پیش فرض (که از جنس فرمول نیز می توانند باشند) از فرم جستجوی پیشرفته دریافت شده (AdvancedSearchEntityArea.GetSearchRepository: 51644c6aa16f) و در مدل فراداده ذخیره می گردد (BizSearchRepository.UpdateAdvancedSearch: 6221f0d749e7). نمایش مقادیر ذخیره شده نیز از طریق متد ShowPreDefinedSearch صورت می گیرد(AdvancedSearchEntityArea.ShowSearchRepository: 8eb8b7ca6c92). از این متد در دوحالت استفاده می گردد، یا در فرم تعریف مقادیر جستجوی تعریف شده در برنامه مدیریت فراداده (به منظور نمایش و اصلاح موارد قبلی) استفاده می گردد که ترکیب مستقیماً به متد نمایش ارسال می شود و یا در برنامه ورود اطلاعات در فرمهای جستجویی که مقادیر پیش فرض معنی دارد به مانند فرم جستجو در گزارشات. در این حالت ترکیب ذخیره شده از طریق پارامتر AdvancedSearchDTOMessage موجود در SearchEntityAreaInitializer (که پیشتر اشاره شده) به منظور نمایش مقادیر پیش فرض در اختیار فرم جستجوی پیشرفته قرار می گیرد.

پس از بررسی زیر فرمهای جستجوی تعریف شده و جستجوی پیشرفته که تشکیل دهنده فرم کلی جستجوی داده می باشند به بررسی سایر کاربردهای فرم جستجوی داده می پردازیم.

### کاربردهای فرم جستجوی داده :

در ابتدا کاربردهای فرم جستجوی داده که دو مورد می باشند بررسی و سپس نحوه نمایش فرم تشریح می گردد.

* مهمترین کاربرد فرم جستجومی داده در همان فرمهای ورود اطلاعات می باشد که در بخشهای قبل توضیحات کاملی درباره آن ارائه شد(BaseEditEntityArea.SearchEntityArea: 54e3b4a36ac9). در این حالت پس از درخواست رویداد جستجو، عملیات جستجو و نمایش داده ها انجام می گیرد (BaseEditEntityArea.SearchEntityArea\_SearchDataDefined: 3841062ffaca)
* کاربرد دیگر در فرم جستجوی عمومی داده می باشد(GeneralEntitySearchArea: 33b533adf734). کاربرد این فرم امکان جستجوی داده ها بصورت کلی می باشد. نمایش نتایج جستجو بستگی به کاربرد این فرم دارد. درواقع در حالتی غیر از جستجو و نمایش داده در فرم های ورود اطلاعات به مانند فرمهای گزارشات و یا نمایش گرافیکی داده ها. در این حالت پس از درخواست رویداد جستجو، عبارت جستجو به فرمهای مصرف کننده جستجوی عمومی فرستاده می شود تا عملیات اختصاصی جستجو و نمایش داده در آنها انجام گردد.

برای نمایش ظاهری فرم جستجو از خصوصیت SearchView در این فرم استفاده می گردد (SearchEntityArea.SearchView: 3100cfe8e22f). بلافاصله بعد از فراخوانی SearchView ظاهر زیر فرمهای جستجوی تعریف شده و جستجوی پیشرفته شده و به این نما افزوده می گردند. مهمترین کاربرد واسط کاربری این فرم به هنگام درخواست نمایش فرم جستجو و نمایش داده در فرمهای ورود اطلاعات می باشد که توسط متد ShowSearchView انجام می گردد(BaseEditEntityArea.ShowSearchView: 397915c7e37f). در این متد زیر فرم جستجوی داده به عنوان بخشی از فرم جستجو و نمایش داده افزوده می گردد. درباره کاربردهای خود فرم نمایش و انتخاب داده در بخش خود توضیحاتی ارائه خواهد شد.

### فرم نمایش و انتخاب داده (ViewEntityArea):

بخش دیگری از فرم جستجو و انتخاب داده در فرم های ورود اطلاعات، زیر فرم نمایش داده های جستجو شده به منظور انتخاب آنها می باشد. برای دسترسی به این زیر فرم در فرمهای ورود اطلاعات از متغیر ViewEntityArea استفاده می گردد(BaseEditEntityArea.ViewEntityArea: 54e3b4a36ac9). این فرم به مانند سایر فرمهای بخش ورود اطلاعات، با ارسال یک ViewEntityAreaInitializer ایجاد می شود. خصوصیات این پارامتر در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| MultipleSelection | bool | حالت چند انتخابی داده ها |
| EntityID | int | شناسه موجودیت |

مبنای این فرم یک گرید و ستونهای نمایش داده های موجودیت مرتبط می باشد. ظاهر فرم در اولن فراخوانی آن ساخته می شود (ViewEntityArea.ViewForViewEntityArea: 14238316d018). در ادامه به بررسی نحوه ساخت فرم می پردازیم.

#### دکمه های فرم نمایش و انتخاب داده :

فرم نمایش و انتخاب داده تنها دارای یک دکمه می باشد که در متد AddCommands افزوده می شود(ViewEntityArea.AddCommands: 0062609d681f). بوسیله آن دکمه کاربر قلم داده های مورد نظر را از جدول داده های نمایش داده شده انتخاب می کند (SelectCommand: 0dd1ef304b03). در واقع با کلیک بروی آن رویداد DataSelected فراخوانی و داده انتخاب شده به فرم ورود اطلاعات مربوطه ارسال می گردد(ViewEntityArea.OnDataSelected: e62db97ed241).

#### دریافت الگوی نمایش و انتخاب داده :

با دسترسی به نمای فرم نمایش و انتخاب داده بلافاصله ظاهر فرم نیز از طریق متد ManageViewEntityArea ساخته می شود(ViewEntityArea.ManageViewEntityArea: d2d99496f04e). در این متد در واقع ستونهای گرید داده تنظیم می گردند. برای افزودن این ستونها به مانند فرم جستجوی تعریف شده نیاز به یک الگو وجود دارد که طبق آن لیست ستونهای نمایشی موجودیت جاری و موجودیت های مرتبط احتمالی به همراه خصوصیات ظاهری آنها مشخص گردد. این الگو از موجودیت EntityListViewDTO ساخته می شود (ViewEntityArea.EntityListView: 890dd76ff4e3) که یک الگوی پیش فرض را برای موجودیت جاری دریافت می کند و یا در صورت عدم وجود آنرا می سازد (BizEntityListView.GetOrCreateEntityListViewDTO: 3462d35bf2fc). ساختار EntityListViewDTO به شرح زیر می باشد:

مبنای اصلی این موجودیت لیستی از EntityListViewColumnsDTO ها می باشد که در واقع همان ستونهای نمایش داده می باشند.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه الگو |
| TableDrivedEntityID | int | شناسه موجودیت مرتبط |
| Title | string | عنوان الگو |
| EntityListViewAllColumns | List< EntityListViewColumnsDTO> itory | لیست ستونها |
| IsDefault | bool | آیا الگوی پیش فرض موجودیت است؟ |

مهمترین خصوصیت آن لیست ستونهای الگو از نوع **EntityListViewColumnsDTO** می باشد که در زیر توصیف شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه ستون نمایش |
| ColumnID | int | شناسه ستون مرتبط |
| Column | ColumnDTO | موجودیت ستون مرتبط |
| OrderID | short | ترتیب ستون |
| WidthUnit | short | عرض ستون |
| Alias | string | عنوان ستون |
| RelationshipPath | string | رشته رابطه |
| RelationshipTail | EntityRelationshipTailDTO | موجودیت رشته رابطه |
| Tooltip | string | توضیحات تکمیلی ستون |
| IsDescriptive | bool | ستون تفسیری موجودیت است؟ |

برای هر موجودیت می توان ترکیب های متفاوت نمایش داده یا به عبارتی EntityListViewDTO های متفاوتی تعریف نمود وسپس از آن در کاربردهای مختلفی به مانند گزارشات استفاده نمود. امّا در فرم های ورود اطلاعات که بحث مورد نظر ما می باشد تنها از الگوی پیش فرض استفاده می گردد که همانطور که ذکر شد این الگو از متد GetOrCreateEntityListViewDTO دریافت می شود. از این متد در استخراج اولیه موجودیت ها از پایگاه داده هدف به منظور ساخت الگوی پیش فرض نیز استفاده می گردد (BizEntitySettings.UpdateDefaultSettingsInModel: 8d0b82740010). قانون افزودن ستونهای ترکیب نمایش پیش فرض در زیر آمده است:

اگر موجودیت نما باشد، تمامی ستونها به عنوان خصوصیت نمایش انتخاب می شوند. در غیر اینصورت به ازای هر ستون قوانین زیر کنترل می شوند(BizEntityListView.GenereateDefaultListViewColumns: e0b50248b309):

* اگر فیلد کلید اصلی باشد به لیست ستونها اضافه می شود.
* اگر فلید کلید خارجی باشد و مربوط به روابط ارث بری فرزند به پدر، چند به یک یا یک به یک صریح باشد
  + ستون به لیست نمایش اضافه می شود.
  + اگر مربوط به رابطه ارث بری باشد، موجودیت طرف رابطه هم برای پیمایش ستونها مجدداً به همین رویداد فرستاده می شود تا این قوانین برای روابط و ستونها آن موجودیت نیز بررسی و در صورت شامل شدن در قوانین به لیست فیلدهای جستجوی موجودیت اولیه اضافه شوند.
  + اگر مربوط به رابطه چند به یک یا یک به یک صریح باشد، اگر در لیست سابقه روابط کمتر از 2 رابطه غیر ارث بری باشد (تنها یک شاخه از همین روابط چند به یک یا یک به یک موجود باشد)، موجودیت طرف رابطه هم مجدداً به همین رویداد فرستاده می شود تا این قوانین برای ستونها و روابط آن موجودیت نیز بررسی و اعمال گردد.
* اگر نام یا عنوان ستون شامل قوانین نام ستونهای نمایش شود به لیست نمایش اضافه می شود(BizEntityListView.CheckColumnDetection: 7a6f5a2ab97e).
* طبیعتاً اگر ستون در بیش از یک قانون شامل شود تنها یکبار به لیست نمایش اضافه می گردد. همچنین اگر ستونی افزوده شود و کلید رابطه باشد (چه کلید اصلی چه خارجی) در بررسی موجودیت طرف دیگر، ستون کلید طرف دیگر افزوده نخواهد شد.

به منظور درک بهتر قوانین ارائه شده در بحث بررسی موجودیت های طرف رابطه اینگونه می توان بیان کرد که زمانی که یک ستونهای یک موجودیت پیمایش می شوند موجودیت های رابطه پدر نیز به مانند موجودیت جاری پیمایش می شوند (زیرا بخشی از موجودیت جاری فرض می گردند) . همچنین در طول هر رشته رابطه مورد بررسی، موجودیتهای روابط چند به یک و یک به یک (صریح) نیز تنها برای یکبار پیمایش می شوند.

#### ساخت الگوی نمایش و انتخاب داده بصورت دستی:

مراحل قبلی توضیحاتی درباره ساخت خودکار الگوی جستجو بود. الگوی نمایش را می توان به صورت دستی نیز در برنامه مدیریت فراداده تعریف نمود. این اتفاق در فرمی مخصوص انجام می گردد(frmEntityListView: a97eb7d1f22e) و سپس الگوی جدید یا موجود در مدل بروزرسانی می گردد (BizEntityListView.UpdateEntityListViews: 738ce2bdd614).

#### ساخت کنترلهای فرم نمایش و انتخاب داده :

بخش اصلی فرم نمایش داده جدولی به منظور نمایش داده ها و شامل ستونهای برگرفته شده از الگو می باشد که در متد ManageViewEntityArea ساخته می شود. در این متد به ازای هر ستون در الگو یک ستون به جدول داده ها افزوده می شود.

### کاربرد فرم نمایش و انتخاب داده :

تنها کاربرد فرم نمایش و انتخاب داده در همان فرم جستجو و نمایش داده در فرمهای ورود اطلاعات می باشد که در بخشهای قبل توضیحات کاملی درباره آن ارائه شد(BaseEditEntityArea.ViewEntityArea: 81a1c5c88a88). در این حالت پس از درخواست انتخاب داده، عملیات افزودن داده انتخاب شده به فرم ورود اطلاعات انجام می گیرد (BaseEditEntityArea.ViewEntityArea\_DataSelected: a377695e6d06)

برای نمایش فرم جستجوی داده از خصوصیت ViewForViewEntityArea آن فرم استفاده می شود (ViewEntityArea.ViewForViewEntityArea: 14238316d018). بلافاصله بعد از فراخوانی ظاهر فرم به ترتیبی که تشریح شد تولید می گردد. امّا موارد استفاده از واسط کاربری این فرم موارد زیر می باشد:

* مهمترین کاربرد واسط کاربری این فرم به هنگام درخواست نمایش فرم جستجو و نمایش داده در فرمهای ورود اطلاعات می باشد که توسط متد ShowSearchView انجام می گردد(BaseEditEntityArea.ShowSearchView: 397915c7e37f). در این متد زیر فرم منایش و انتخاب داده به عنوان بخشی از فرم جستجو و نمایش داده افزوده می گردد. درباره کاربردهای خود فرم نمایش و انتخاب داده در بخش خود توضیحاتی ارائه خواهد شد.
* این فرم می تواند بصورت Popup نیز در فرمهایی غیر مستقیم به منظور میانبری برای جستجو و نمایش و انتخاب داده ها استفاده گردند(به مانند فرم لوکاپی) (BaseEditEntityArea.ShowTemproraryUIViewArea: 0cfe0a61872b)

پس از تشریح کامل فرم جستجو و انتخاب داده و همچنین زیر فرمهای مربوطه (فرم جستجو و فرم نمایش و انتخاب داده) به عنوان مبحث پایانی این بخش به تشریح کاربردهای فرم جستجو و انتخاب داده می پردازیم.

## کاربردهای فرم جستجو وانتخاب داده :

فرم جستجو و نمایش و انتخاب داده خود کاربردی عملیاتی ندارد زیرا عملیاتهای اصلی آن در زیر فرم های مستقل جستجو و نمایش داد انجام می گردد. تنها کاربرد آن واسط کاربری فرم می باشد که در خصوصیت ViewForSearchAndView فرم های ورود اطلاعات قابل دسترسی است (ViewForSearchAndView: f3b7615955e1). عمده استفاده از این واسط کاربری در متد ShowSearchView می باشد همان جایی که این واسط کاربری برای اولین بار ساخته می شود و در پی آن واسط کاربری ها زیر فرم های جستجو و زیر فرم نمایش داده ساخته شده و به آن افزوده می گردد (BaseEditEntityArea.ShowSearchView: 397915c7e37f). هدف از متد ShowSearchView یکپارچه کردن فراخوانی ها ممکن به منظور نمایش فرم جستجو و نمایش داده در فرم های ورود اطلاعات می باشد. در در بخش زیر به کاربردهای این نیازمندی پرداخته می گردد:

* با کلیک بروی دکمه جستجو در فرم های مستقیم ورود اطلاعات (SearchCommand.SearchCommand: c971552116e1)
* با کلیک بروی دکمه جستجوی فرم های غیر مستقیم به عنوان فرم اصلی (BaseEditEntityArea.TemporaryDisplayView\_TemporaryDisplayViewRequested: 95a95d63fa08)
* با کلیک بروی دکمه جستجوی فرم های غیر مستقیم حاصل از رابطه (ChildRelationshipInfo.TemporaryViewActionRequested: 40b8056e73af)

با اتمام تشریح مفاهیم ساخت و زیر فرمهای یک فرم ورود اطلاعات، در ادامه به کاربردهای فرمهای ورود اطلاعات پرداخته می شود.

## کاربردهای فرم کار با موجودیت داده :

همانطور که در ابتدای مبحث کار با فرم ورود اطلاعات اشاره شد، برای ایجاد یک فرم ورود اطلاعات از متد عمومی GetEditEntityArea استفاده می شود. بنابراین برای بررسی کاربردهای فرمهای ورود اطلاعات کافی است تا کاربردها و نحوه تعامل با این متد بررسی شود. انواع حالتهای ایجاد فرم ورود اطلاعات را می توان به سه دسته تقسیم نمود که در زیر به آنها اشاره شده است:

1. ایجاد و نمایش یک فرم مستقیم ورود اطلاعات به عنوان پنجره اصلی که به آن پنجره اصلی ورود اطلاعات اطلاق می کنیم.
2. ایجاد فرم غیر مستقیم به عنوان فرم اولیه صرفاً به منظور انتخاب تک داده که به آن برای سهولت فرم انتخاب تک داده اطلاق می کنیم.
3. ایجاد فرم ورود اطلاعات بنابر شرایط خاص و پارامترهای ارسالی گوناگون که به آن فرم اختصاصی اطلاق می کنیم.

در ادامه به نقاطی که متد GetEditEntityArea به منظور ایجاد فرم فراخوانی می شود بر اساس دسته بندی ذکر شده می پردازیم:

### پنجره اصلی ورود اطلاعات:

برای ایجاد فرمهای ورود اطلاعات از متد ShowEditEntityArea استفاده می گردد (AgentUICoreMediator.ShowEditEntityArea: cc973c4f9fdb). این متد یک فرم ورود اطلاعات مستقیم را در یک پنل بزرگ نمایش خواهد داد. همچنین می توان داده یا داده هایی را به عنوان داده اولیه فرم معرفی نمود تا پس از ایجاد فرم بلافاصله در فرم نمایش داده شوند. کاربردهای این روش ایجاد فرم ورود اطلاعات موارد زیر می باشند:

درخت منو:

زمانی که بروی منوی فرمهای برنامه ورود اطلاعات کلیک می شود فرم ورود اطلاعات مرتبط نمایش داده می شود. در این حالت هیچ داده پیش فرضی به فرم ارسال نمی گردد (AgentUICoreMediator.NavigationMenuClicked: a3a9f1a1fdd4)

#### منوی کلیک راست داده:

زمانی که در منوی داده بروی نمایش فرم کلیک می گردد. در این حالت داده جاری به عنوان داده پیش فرض نمایش در فرم ارسال می گردد (DataMenuArea.MenuUI\_MenuClicked: 28b138367084)

#### کارتابل و نمایش فرم رشته رابطه :

در کارتابل زمانی که فرمی بر اساس رشته رابطه مرتبط با داده اصلی معرفی شده است. داده های هدف رشته رابطه مرتبط با داده اصلی به عنوان داده پیش فرض به فرم ارسال می گردند (CartableArea.MnuEntity\_MenuItemClicked: 8fea5880e223)

### فرم انتخاب تک داده:

مواردی که تنها انتخاب داده به عنوان فرم اولیه مورد نیاز است کاربردهای متعددی دارد. در جدول زیر تنها به این کاربردها اشاره شده و در بخش های مرتبط به این تشریح این کاربردها بیشتر پرداخته خواهد شد.

|  |  |
| --- | --- |
| عنوان کاربرد | توضیحات |
| فرم تعریف فرمول | برای تست فرمول تعریف شده در یک فرم ورود اطلاعات نمونه (frmNewFormulaDefinition.SetEditEntityArea: 909072457fda) |
| لینک داده | در بخش لینک داده زمانی که فرم انتخاب داده مورد نیاز است (DataLinkArea.SetDataLink: cdff2b2126ea) |
| فرم عمومی انتخاب داده | در فرم عمومی انتخاب داده زمانی که موجودیت مورد نظر مشخص می شود (EntitySelectArea.EntitySearchLookup\_SelectionChanged: c3861e5f6f81). این فرم عمومی در بخشهای آرشیو و نامه و گزارشات مستقیم کاربرد دارد که در بخشهای مربوطه به آنها پرادخته می شود. |
| فرم گراف داده | در فرم گراف داده زمانی که نیاز است یک فرم انتخاب داده نمایش داده شود (GraphArea.SetGraph: d3b96a0f7c64) |
| فرم گزارش لاگ | در فرم لاگ داده زمانی که نیاز است یک فرم انتخاب داده نمایش داده شود (LogReportArea.EntitySearchLookup\_SelectionChanged: 6e317f93bebd) |
| فرم ایجاد جریان کار | در فرم ایجاد جریان کار زمانی که نیاز است یک فرم انتخاب داده نمایش داده شود (WorkflowCreateRequestArea.View\_ProcessSelected: 016da5fcba02) |
| فرم گزارش جریان کار | در فرم گزارش جریان کار زمانی که نیاز است یک فرم انتخاب داده نمایش داده شود (WorkflowReportArea.AddDataSelector: c283f53e955b) |
|  | بهتره این انتخاب داده ها همگی یکسان سازی بشن مثلاً مثل انتخاب داده در ارشیو و نامه |

### ایجاد فرم اختصاصی

این حالت از ایجاد فرم حالت مشخصی از منظر نوع تعامل و یا چند داده ای بودن فرم ندارد و بنابر مورد کاربرد این پارامترها می توانند متفاوت باشند.

#### نمایش رابطه در فرم ورود اطلاعات

زمانی که در یک فرم ورود اطلاعات نیاز است تا طرف دیگر رابطه بصورت فرم نمایش داده شود و نوع ایجاد فرم به تنظیمات رابطه بستگی دارد (BaseEditEntityArea.GenerateRelationshipControlEditArea: 7b5673de233c)

#### فرم نمونه در بخش تعریف الگوی فرمها:

در برنامه ورود اطلاعات و در فرمی که الگوهای فرمهای ورود اطلاعات را تعریف می کنیم نیاز است تا نتیجه کار بصورت نمونه به کاربر نمایش داده شود (frmEntityUIComposition.btnPreview\_Click: 942c5c707156)

#### فرم جستجوی از **پیش** تعریف شده

در فرم جستجوی تعریف شده نیاز است تا یک فرم غیر مستقیم امّا بصورت چند داده ای به منظور انتخاب داده های فیلتر جستجو انتخاب شوند (EntityDefinedSearchArea.GenerateUIControls: 31d456f709e0)

## تعامل فرمها و داده ها

تا به اینجا بیشتر از ساختار ایستای فرمهای کار با داده صحبت به میان آمد در حالیکه این فرمها بدون داده و عملکردهای متناسب با داده ها هیچگونه کاربردی ندارند. در ادامه به نحوه تعامل فرمها با داده می پردازیم. در ابتدا حالت مدنظر فرم ما همان پنجره اصلی ورود اطلاعات می باشد. با بررسی این حالت نکات کلیدی بسیاری از حالتهای دیگر فرمی که پیشتر اشاره شدند پوشش داده خواهد شد.

همانطور که اشاره شد برای ایجاد پیجره ورود اطلاعات از متد ShowEditEntityArea استفاده می شود. در این حالت یک فرم ورود اطلاعات مستقیم از موجودیت مشخص شده ایجاد می گردد(از طریق GetEditEntityArea). سپس پیش از نمایش فرم، متد ShowDataFromExternalSource فراخوانی می گردد و داده های پیش فرض (در صورت وجود) به آن ارسال می گردند(BaseEditEntityArea.ShowDataFromExternalSource: 0594c59749b4).

### متد نمایش داده های ارسالی از خارج فرم (ShowDataFromExternalSource):

اساساً کارکرد این متد نمایش داده های در یک فرم کار با داده می باشد زمانی که داده ها از بخشی خارج از فرم به آن ارسال می شوند. این متد در حالتهای دیگر فرم های کار با داده به مانند حالت فرم مستقل انتخاب تک داده کاربرد دارد. لیست داده ورودی آن از نوع DP\_BaseData می باشد که ابتدایی ترین ساختار داده در برنامه می باشد(DP\_BaseData: ffcd7c3d2304) و شاکله اصلی آن شناسه موجودیت مرتبط و لیستی از خصوصیتها و مقادیر مربوطه می باشد. به بررسی تخصصی تر انواع داده در بخشهای بعدی خواهیم پرداخت، نکته مهم در اینجا ارسال داده های پیش فرض به متد نمایش داده خارجی می باشد که این لیست می تواند دارای قلم داده باشد و یا یک لیست خالی (برای حالتهایی که داده پیش فرضی وجود ندارد). در دو حالت رویدادهای زیر اتفاق می افتد :

* **اگر داده ای وجود داشت:**
  + اگر فرم در حالت مستقیم یا همان ورود اطلاعات ایجاد شده باشد: داده ها از طریق متد SearchDataForEditFromExternalSource و با استفاده از مقادیر کلید های اصلی از حالت DP\_BaseData به حالت داده فرمی یا همان DP\_FormDataRepository تبدیل می شوند
  + اگر فرم در حالت غیر مستقیم ایجاد شده باشد: داده ها از طریق متد SearchDataForViewFromExternalSource و با استفاده از مقادیر کلید های اصلی از حالت DP\_BaseData به حالت داده فرمی یا همان DP\_FormDataRepository تبدیل می شوند
  + داده های تبدیل شده از طریق متد AddData به فرم افزوده می گردند.
* **اگر داده ای وجود نداشته باشد**
  + اگر فرم تک داده باشد یک داده پیش فرض خالی از طریق متد CreateDefaultData ایجاد می گردد که در دل آن همان داده پیش فرض از طریق متد AddData به فرم افزوده می گردد.

در توضیحات ارائه شده به چند مفاهیم و عملکردهایی از جمله داده فرمی، تبدیل داده، متد AddDate و CreateDefaultData اشاره شد. در ادامه به تشریح و کاربرد هر کدام از این مفاهیم پرداخته می شود.

### انواع داده در برنامه :

نوع داده ای که پیشتر اشاره شد نوع داده پایه یا همان DP\_BaseData بود. این نوع داده برای تعامل با پیچیدگیهای فرمهای ورود اطلاعات بسیار ساده می باشد. به این منظور انواع گوناگون داده طراحی شده اند که هر کدام بنابر سطح کاربردی متفاوت از داده های پایه تر مشتق شده اند. در ادامه به کل انواع داده و کاربردهای آنها می پردازیم.

#### نوع داده پایه :

نوع پایه داده (DP\_BaseData: ffcd7c3d2304) که شاکله اصلی آن شناسه موجودیت مرتبط (TargetEntityID) و لیستی از خصوصیتها از نوع EntityInstanceProperty می باشد که حاوی اطلاعات ستون و مقادیر آنهاست(EntityInstanceProperty: 3b18025dd763). به هر خصوصیت (EntityInstanceProperty)، خصوصیت داده اطلاق می کنیم که ساختار آن در زیر آمده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ColumnID | int | شناسه ستون مرتبط |
| Column | ColumnDTO | موجودیت ستون مرتبط |
| Value | object | مقدار خصوصیت |
| IsKey | bool | آیا ستون کلید اصلی است؟ |
| FormulaID | int | شناسه فرمول مرتبط |
| ValueIsChanged | int | آیا مقدارش تغییر کرده است؟ |
| RelationshipIDTailPath | int | شناسه رشته رابطه |
| HasForeignKeyData | int | آیا مقدار کلید خارجی دارد؟ |
| ISFK | int | آیا ستون کلید خارجی است؟ |
| ListViewColumn | int | موجودیت ستون نمایش مرتبط |
| ValueIsEmpty | int | آیا مقدارش خالی است؟ |

مهمترین خصوصیاتی که در بخشهای جاری با آنها کار داریم همان ستون مرتبط و مقدار هر خصوصیت داده می باشد. بررسی سایر خصوصیت را در کاربردشان در بخشهای آتی انجام خواهیم داد.

#### نوع داده نمایشی:

* این نوع به نام DP\_DataView می باشد و از داده پایه مشتق می شود(DP\_DataView: 4a4b577125bf) و کاربرد آن زمانی است که داده فقط خواندنی با ستونهای تفسیری مناسب مورد نیاز است. به مانند داده های فرم نمایش داده و یا داده های انتخاب شده در فرمهای غیر مستقیم کار با داده. خصوصیات این داده در زیر اشاره شده اند:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ListViewID | List<int> | شناسه لیست نمایش |
| DataViewProperties | List<DataViewProperty> | خصوصیات نمایشی |
| ViewInfo | string | رشته نمای داده |

* **خصوصیت DataViewProperties:** لیست خصوصیات نمایشی داده در این خصوصیت نگهداری می شود. هر DataViewProperty شامل دو خصوصیت مهم است، یکی خصوصیت ListViewColumn که ستون نمایشی مرتبط است و دیگری خصوصیت Value که مقدار خصوصیت است.
* **رشته نمای داده ViewInfo :** هر داده دارای یک رشته خلاصه اطلاعات برای نمایش در صورت نیاز در واسط کاربری می باشد. در واقع مشخصه ای که عنوان داده را توصیف می کند.

کلاً این بخش پس از تثبیت برنامه توضیحات کامل تری ارائه شود

#### نوع داده داینامیک:

این نوع داده به نام DP\_DataRepository می باشد و از نوع داده نمایشی مشتق می شود (DP\_DataRepository: 7939d86acdc6) هدف اصلی آن تعریف داده های مرتبط طبق الگوی فرمهای مرتبط و تودرتو می باشد. قابلیتهای آن امکان تعریف و اصلاح لیست خصوصیات، تعریف داده مرتبط، نگهداری خصوصیات اصیل داده و ... می باشد. یکی از کاربردهای اصلی این نوع داده نقل و انتقال قلم داده ها بین واسط کاربری و سرویسهای CRUD پایگاه داده هدف می باشد. خصوصیات و متد های مهم این نوع داده در زیر اشاره شده اند:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ChildRelationshipDatas | List<ChildRelationshipData> | لیست داده-رابطه های زیر مجموعه |
| ParantChildRelationshipData | ChildRelationshipData | رابطه-داده پدر |
| OriginalProperties | List<EntityInstanceProperty> | خصوصیات اصیل داده |
| IsFullData | bool | آیا داده کامل است؟ |
| EntityListView | EntityListViewDTO | لیست نمایش مرتبط |
| DataView | DP\_DataView | نمای داده |
| ViewInfo | string | رشته نمای داده |

* **لیست رابطه داده های مرتبط (ChildRelationshipDatas):** اینکه برای داده موجود (که به یک فرم ورود اطلاعات اختصاص دارد) داده های فرم های رابطه های زیر مجموعه آن کدام داده ها می باشند. در واقع لیستی از ChildRelationshipData ها که ساختار آن در زیر آمده (ChildRelationshipData: 247b7f38d6a7) :

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| SourceData | DP\_DataRepository | داده مبداً رابطه |
| Relationship | RelationshipDTO | رابطه از مبداً به طرق دیگر رابطه |
| OriginalRelatedData | List<DP\_DataRepository> | داده های اصیل طرف دیگر رابطه (مقصد) |
| RelatedData | List<DP\_DataRepository> | داده های مرتبط طرف دیگر رابطه (مقصد) |
| RelationshipIsChangedForUpdate | bool | آیا رابطه برای بروزرسانی تغییر کرده |
| RemovedDataForUpdate | List<DP\_DataRepository> | داده های تغییر کرده |

مهمترین خصوصیات مورد نظر در این بخش همان داده مبداً (SourceData یا همان داده فرم جاری)، رابطه از مبداً به فرم مقصد(Relationship) و داده های فرم مقصد یا همان فرم مجموعه مرتبط (RelatedData) می باشد.

* **رابطه-داده پدر (ParantChildRelationshipData):** اگر خود داده از یک زیر بوجود آمده باشد این خصوصیت مقدار می گیرد و به رابطه داده از فرم پدر به فرم جاری اشاره دارد. نوع آن هم همان ChildRelationshipData است که داده پدر یا مبدأ را به داده جاری متصل کرده.
* **خصوصیات اصیل (OriginalProperties):** هر خصوصیتی که به داده اضافه می شود یک نسخه از آن در این خصوصیا نگهداری می شود تا مقادیر اولیه خصوصیت نگهداری شود. کاربرد آن در ایجاد کوئری های بروزرسانی داده ها و یا ثبت لاگ تغییرات داده می باشد.
* **خصوصیت IsFullData :** این خصوصیت نشان دهنده اینست که آیا داده برای یک فرم مستقیم ورود اطلاعات ایجاد یا تبدیل شده است یا خیر. در واقع آیا تمام خصوصیات مورد نیاز برای فرم ورود اطلاعات برای داده موجود می باشند. در این صورت به آن داده کامل اطلاق می کنیم.
* **رشته نمای داده ViewInfo :** هر داده دارای یک رشته خلاصه اطلاعات برای نمایش در صورت نیاز در واسط کاربری می باشد. در واقع مشخصه ای که عنوان داده را توصیف می کند.

#### نوع داده فرمی:

این نوع داده از داده داینامیک مشتق می شود و هدف از ایجاد آن تعریف داده ای است در تعامل با واسط کاربری فرمهای کار با داده. در واقع رابطی بین داده داینامیک و کنترلهای واسط کاربری. خصوصیات مهم این نوع داده در زیر آمده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ChildSimpleContorlProperties | List<ChildSimpleContorlProperty> | داده مبداً رابطه |
| ChildRelationshipDatas | List<ChildRelationshipInfo> | رابطه از مبداً به طرق دیگر رابطه |
| ISValid | bool | داده های اصیل طرف دیگر رابطه (مقصد) |
| ShoudBeCounted | bool | داده های مرتبط طرف دیگر رابطه (مقصد) |
| DataOrRelatedDataIsChanged | bool | آیا رابطه برای بروزرسانی تغییر کرده |
| IsDefaultData | bool | داده های تغییر کرده |
| IsUpdated | bool | داده های تغییر کرده |
| ListTempSimplePropertyReadonly | List<Tuple<int, string, string, bool>> | داده های تغییر کرده |
| ListTempRelationshipPropertyReadonly | List<Tuple<int, string, string, bool>> | داده های تغییر کرده |
| ToParentRelationshipIsReadonly | List<ControlStateItem> | داده های تغییر کرده |
| ToParentRelationshipIsHidden | List<ControlStateItem> | داده های تغییر کرده |

### تبدیل انواع داده:

همانطور که در بخش قبل انواع داده تشریح شدند، مشخص است که هر نوع داده کاربرد خاص خود را دارند. برای مثال برای یک فرم غیر مستقیم در مرحله اول نیازی نیست تا یک داده کامل با تمام خصوصیات و روابط زیر مجموعه ساخته و یا از پایگاه داده دریافت شود، امّا اگر در همان فرم دکمه نمایش فرم ورود اطلاعات کلیک شد، نیاز است تا آن داده به یک داده کامل تبدیل شده و در فرم به همراه تمامی خصوصیات و روابط نمایش داده شود. یا در مثالی دیگر در همان متد ذکر شده نمایش داده از خارج فرم (ShowDataFromExternalSource) داده ورودی به صورت پایه و احتمالاً تنها شامل مقادیر ستونهای کلید اصلی می باشد بنابراین در اولین گام نیاز به تبدیل داده به یک داده فرمی می باشد. حال اگر فرم مستقیم بود داده فرمی کامل و در غیر اینصورت داده فرمی از نمای داده تهیه می گردد.

انواع تبدیل داده :

.....

### افزودن داده به فرم (متد AddData) :

از این متد برای افزودن و نمایش داده ها در فرم اصلی استفاده می شود. اگر فرم اصلی فرم مستقیم باشد داده از طریق متد ShowDataInDataView نمایش هم داده می شود. اگر فرم غیر مستقیم باشد خلاصه اطلاعات داده های موجود از طریق متد SetTempText نمایش داده می شود. کاربرد این متد در چهار مورد می باشد :

* زدن دکمه افزودن (AddCommand) در فرم اصلی
* افزودن داده ارسال شده از خارج فرم بعد از تبدیل. در واقع همان متد ShowDataFromExternalSource که پیشتر اشاره شد.
* در متد AddDataViewItemsToForm (بررسی و اصلاح شود)
* به هنگام ایجاد داده خالی پیش فرض برای فرم ورود اطلاعات تک داده از طریق متد CreateDefaultData

سایر نکات متد AddData

### ایجاد داده خالی پیش فرض برای فرم ورود اطلاعات تک داده (CreateDefaultData) :

در صورتی که نیاز به نمایش فرم ورود اطلاعات تک داده باشد و داده ای برای نمایش وجود نداشته باشد بوسیله متد CreateDefaultData یک داده پیش فرض با مقادیر خالی اولیه ایجاد می گردد و به فرم به منظور نمایش افزوده می شود(EditEntityAreaOneData.CreateDefaultData: f089224255d6). به منظور ایجاد داده پیش فرض از متد CreateAreaInitializerNewData استفاده می شود. کاربرد های متد CreateDefaultData موارد زیر می باشد :

* ایجاد داده پیش فرض برای فرم اصلی تک داده مستقیم زمانی که داده خارجی ارسال نشده است. در همان متد ShowDataFromExternalSource که پیشتر اشاره شد.
* زمانی که داده جاری از فرم ورود اطلاعات حذف می گردد و نیاز است که مجدداً داده پیش فرضی نمایش داده شود.

لازم به توضیح است که این متد خاص مخصوص فرم اصلی ورود اطلاعات می باشد و برای روابط داده نیز در صورت نیاز متد مشابهی موجود می باشد که در بخشهای بعد به آن پرداخته خواهد شد.

### ایجاد داده پیش فرض (CreateAreaInitializerNewData):

برای ایجاد خود داده از متد CreateAreaInitializerNewData استفاده می شود(AgentHelper.CreateAreaInitializerNewData: 193cdabb907a). در این متد داده کاملی برای موجودیت فرم جاری ایجاد شده و به ازای تمامی کنترلهای ستونها و روابطش فرم جاری خصوصیات پیش فرض با مقادیر پیش فرض اولیه به داده اضافه می شوند. کاربردهای این متد موارد زیر می باشند:

* با زدن دکمه افزودن (AddCommand) در فرم ورود اطلاعات نیاز است تا یک داده پیش فرض ایجاد شده و به فرم اصلی یا رابطه جاری فرم افزوده شود.
* به هنگام ایجاد داده پیش فرض فرم اصلی تک داده که در بخش قبل توضیح داده شد.
* به هنگام ایجاد داده پیش فرض فرم رابطه تک داده.

در جریان تشریح متد ShowDataFromExternalSource یا همان نمایش داده خارجی بودیم که نیاز شد تا برخی متدهای پایه فرمهای اصلی معرفی گردند. همانطور که ذکر شد، پس از ارسال داده های پایه خارجی به این متد عملیات تبدیل داده انجام می شود. برای تبدیل در فرمهای مستقیم که نیاز به داده کامل است از متد SearchDataForEditFromExternalSource و برای فرمهای غیر مستقیم که نیاز به داده نمایشی است از متد SearchDataForViewFromExternalSource استفاده می شود. در ادامه به بررسی اختصاصی این دو متد می پردازیم.

#### تبدیل داده خارجی به داده کامل برای فرم مستقیم (SearchDataForEditFromExternalSource) :

عملکرد اصلی متد SearchDataForEditFromExternalSource، دریافت یک داده پایه (BaseData) و تبدیل آن به داده فرمی کامل از طریق پرس و جو از پایگاه داده هدف می باشد (BaseEditEntityArea.SearchDataForEditFromExternalSource: 12958ab8fff9). برای داده خارجی وجود خصوصیات کلید پایه ضروری می باشد. از طریق همین خصوصیات یک درخواست جستجو از نوع DR\_SearchEditRequest ساخته می شود و به متد دریافت داده طبق جستجوی ارسالی فرستاده می شود (SendSearchEditRequest (RequestManagerService.SendSearchEditRequest: a0a0ef868831). خروجی این متد از نوع داده داینامیک می باشد. سپس طبق این نتایج داده های فرمی مربوطه در همان متد SearchDataForEditFromExternalSource ساخته شده و برگردانده می شود.

#### تبدیل داده خارجی به داده نمایشی برای فرم غیر مستقیم (SearchDataForViewFromExternalSource) :

عملکرد اصلی متد SearchDataForViewFromExternalSource، دریافت یک داده پایه (BaseData) و تبدیل آن به داده نمایشی از طریق پرس و جو از پایگاه داده هدف می باشد (BaseEditEntityArea.SearchDataForViewFromExternalSource: 7a04c6c350ac). برای داده خارجی وجود خصوصیات کلید پایه ضروری می باشد. از طریق همین خصوصیات یک درخواست جستجو از نوع DR\_SearchViewRequest ساخته می شود و به متد دریافت داده طبق جستجوی ارسالی فرستاده می شود (RequestRegistrationService.SendSearchViewRequest: 29b1b425fff4). خروجی این متد از نوع داده نمایشی می باشد. سپس طبق این نتایج داده های فرمی مربوطه در همان متد SearchDataForViewFromExternalSource ساخته شده و برگردانده می شود.

در اینجا مبحث نمایش داده خارجی در فرمهای اصلی به پایان می رسد. همانطور که در ابتدای بخش اصلی جاری اشاره شد فرایند نمایش داده خارجی در ادامه نمایش پنجره فرم اصلی در متد ShowEditEntityArea اجرایی می گردید. زیر فرایند اصلی که در این فرایندهای نمایش داده خارجی انجام گردید افزودن داده های فرمی بدست آمده از طریق متد AddData به فرم اصلی برنامه بود. همانطور که در توضیح متد AddData اشاره شد بخش مهمی از آن نمایش اطلاعات داده چه برای فرم مستقیم و چه فرم غیر مستقیم می باشد. در ادامه به فرایند نمایش داده های فرمی در یک فرم (که اینجا فرم اصلی می باشد) می پردازیم.

### نمایش داده در فرم ورود اطلاعات (ShowDataInDataView):

وظیفه اصلی این متد نمایش یک داده فرمی در واسط کاربری فرم می باشد (BaseEditEntityArea.ShowDataInDataView: 9f08862e5c0d). به عبارتی تمامی ستونهای داده فرمی به کنترلهای ستون و تمامی روابط داده به کنترلهای رابطه متصل (Bind) می گردند. به طور دقیق رویدادهای زیر در این متد انجام می گردند:

* اگر فرم چند داده ای باشد یک ردیف به جدول مدیریت داده ها در واسط کاربری فرم افزوده می گردد (View\_MultipleDataContainer.AddDataContainer: 30bce92856e7)
* به ازای تمامی خصوصیات ساده داده (خصوصیات موجود در ChildSimpleContorlProperties) متد SetBinding فراخوانی می شود
* به ازای تمامی خصوصیات رابطه داده (خصوصیات موجود در ChildRelationshipDatas) متد SetBinding فراخوانی می شود

کاربردهای این متد دو مورد می باشند:

* کاربرد اول در همان متد AddData فرم اصلی است که به آن اشاره شد.
* کاربرد دیگر زمانی است که نیاز است داده های رابطه نمایش داده شوند (ChildRelationshipInfo. ShowDataInChildRelationshipDataView: 78184401047a).

حال بررسی کنیم که عملیاتهای SetBinding بروی خصوصیت ساده و رابطه چه تاثیری دارند.

#### عملیات SetBinding بروی خصوصیات ساده :

عملیات اتصال خصوصیت ساده به کنترل مربوطه در واسط کاربری از طریق این متد انجام می گردد(ChildSimpleContorlProperty.SetBinding: 9bef1c5de838). این عملیات شامل دو مرحله می شود، مرحله اول بایند شدن خصوصیت به کنترل مربوطه از طریق واسط مدیریت کنترل (I\_UIControlManager: c14447d93698) و سپس انجام تنظیمات ظاهری و کنترلی بروی کنترل (ChildSimpleContorlProperty.SetControlUIDetails: ffee3df3-b262-4bb8-88ba-2b095307e547) که درباره جزئیات آن در بخشهای بعدی صحبت خواهد شد.

#### عملیات SetBinding بروی خصوصیات روابط :

عملیات اتصال خصوصیت رابطه به کنترل مربوطه در واسط کاربری از طریق این متد انجام می گردد(ChildRelationshipInfo.SetBinding: b0d3af993f9c). این عملیات به علت درگیر بودن رابطه و داده های مربوطه و همچنین ارتباط با فرم طرف دیگر رابطه دارای جزئیات و زیر عملیاتهایی می باشد. روال انجام عملیات درزیر اشاره شده:

* برای فرم مقصد تعیین می شود که خصوصیت-رابطه جاری فرم همین خصوصیت مورد نمایش می باشد.
* اگر فرم طرف دیگر مستقیم بود، ظاهر فرم خالی می شود. به عبارتی تنها برای فرمهای چند داده ای رکوردهای موجود در جدول داده حذف می شوند (ChildRelationshipInfo.ClearUIData: a9614860d724)(تنها حذف ظاهری و تاثیری در چینش داده ها ندارد). اگر فرم غیر مستقیم بود بخش خلاصه داده خالی می گردد(UC\_TemporaryDataSearchLink.SetLinkText: 9168b4b9155f).
* اگر داده های رابطه قبلاً از پایگاه داده هدف جستجو شده نشده باشند، یا در واقع داده کامل پدر برای اولین بار در فرم ورود اطلاعات نمایش داده شده باشد :
  + اگر ستونهای طرف اول رابطه در داده پدر دارای مقدار باشند
    - اگر فرم رابطه مستقیم باشد، داده های فرمی کامل طرف دیگر رابطه از طریق متد SerachDataFromParentRelationForChildDataView دریافت می شوند(ChildRelationshipInfo.SerachDataFromParentRelationForChildDataView: 847beee04f15)
    - اگر فرم رابطه غیر مستقیم باشد، داده های فرمی نمایشی طرف دیگر رابطه از طریق متد SerachDataFromParentRelationForChildTempView در یافت می شوند(ChildRelationshipInfo.SerachDataFromParentRelationForChildTempView: 0f1244a3531e).
    - اگر مشکلی در دریافت داده ها نبود، هر داده از طریق متد AddDataToChildRelationshipData به رابطه افزوده شده و نمایش داده خواهد شد.
* اگر داده های رابطه قبلاً یکبار جستجو شده باشند یا به عبارتی داده کامل پدر برای اولین بار در فرم ورود اطلاعات نمایش داده نشده باشد :
  + اگر فرم مستقیم باشد، هر داده تک به تک در فرم رابطه نمایش داده خواهد شد (ChildRelationshipInfo. ShowDataInChildRelationshipDataView: 78184401047a)
  + اگر فرم غیر مستقیم باشد، خلاصه اطلاعات داده ها در فرم غیر مستقیم نمایش داده خواهد شد (ChildRelationshipInfo. SetTempTextChildRelationshipData: 2da7f695849f)
* در ادامه و پس از مرحله دریافت و نمایش داده ها، اگر فرم مستقیم و تک داده ای باشد و اگر داده ای موجود نباشد در رابطه، داده پیش فرض از طریق متد CreateDefaultDataChildRelationship ایجاد می گردد.
* در آخر تنظیمات ظاهری رابطه و فرم مربوطه انجام می گردند.

در تشریح روال عملیات اتصال روابط، به برخی متدها به مانند متدهای تبدیل داده و یا AddDataToChildRelationshipData و CreateDefaultDataChildRelationship اشاره شد که گرچه مشابه آنها در تشریح نمایش داده خارجی در فرم اصلی (ShowDataFromExternalSource) وجود داشت امّا تفاوتهایی عملیاتی دارند. در ادامه به تشریح این متدهای اختصاصی در بحث مدیریت روابط و فرمهای مربوطه می پردازیم.

#### متد دریافت داده کامل برای رابطه مستقیم:

برای دریافت داده های طرف دیگر رابطه زمانی که فرم طرف دیگر مستقیم می باشد از متد SerachDataFromParentRelationForChildDataView استفاده می کنیم(ChildRelationshipInfo.SerachDataFromParentRelationForChildDataView: 847beee04f15). در این متد عبارت جستجو با توجه به موجودیت طرف دوم و فیلدهای رابطه از طرف اول ساخته شده و نهایتاً داده ها از طریق SendSearchEditRequest (RequestManagerService.SendSearchEditRequest: a0a0ef868831) دریافت می شوند. معادل این متد برای فرم اصلی ورود اطلاعات همان متد SearchDataForEditFromExternalSource می باشد.

#### متد دریافت داده کامل برای رابطه غیر مستقیم:

برای دریافت داده های طرف دیگر رابطه زمانی که فرم طرف دیگر مستقیم می باشد از متد SerachDataFromParentRelationForChildTempView استفاده می کنیم ChildRelationshipInfo.SerachDataFromParentRelationForChildTempView: 0f1244a3531e. در این متد عبارت جستجو با توجه به موجودیت طرف دوم و فیلدهای رابطه از طرف اول ساخته شده و نهایتاً داده ها از طریق SendSearchViewRequest (RequestRegistrationService.SendSearchViewRequest: 29b1b425fff4) دریافت می شوند. معادل این متد برای فرم اصلی ورود اطلاعات همان متد SearchDataForViewFromExternalSource می باشد.

### متد افزودن داده به رابطه (AddDataToChildRelationshipData):

در بحث افزودن داده ها مرتبط رابطه مستقیم به هنگام نمایش داده پدر در فرم ورود اطلاعات به این متد اشاره شد. عملکرد این متد به مانند متد AddDate در فرم اصلی می باشد منتها این بار برای یک خصوصیت رابطه (ChildRelationshipInfo.AddDataToChildRelationshipData: 6e846e90a6ff). فرایندی که در این متد طی می شود در زیر اشاره شده است:

* برای داده هدف خصوصیت رابطه پدر یا همان ParantChildRelationshipData مقداردهی می شود، برابر با خصوصیت-رابطه جاری
* داده به لیست روابط خصوصیت-رابطه یا همان خصوصیت RelatedData افزوده می شود.
* اگر داده های رابطه مستقیماً از پایگاه داده خوانده شده باشند، آنگاه به عنوان داده های اصیل خصوصیت-رابطه نگهداری می شوند.
* اگر داده ها مربوط به پایگاه داده نباشند و در فرم مرتبط شده باشند آنگاه مقادیر خصوصیات کلید خارجی در صورت نیاز توسط متد CheckRelationshipChanged مقداردهی می شود. در بخشهای بعدی به این متد اشاره خواهد شد.
* اگر فرم ورود اطلاعات رابطه باز باشد، یعنی فرم مستقیم باشد و یا فرم ورود اطلاعات برای حالت غیر مستقیم باز باشد (ChildRelationshipInfo.IsDataviewOpen: fb8b8d086bbe)، داده از طریق متد ShowDataInDataView در فرم خصوصیت-رابطه مربوطه نمایش داده می شود.
* اگر فرم غیر مستقیم باشد اطلاعات خلاصه داده ها در فرم غیر مستقیم نیز بروزرسانی می شود.
* تنظیمات ظاهری فرم رابطه انجام می شود.

امّا متد افزودن داده تنها مختص اتصال داده های رابطه از فرم پدر به فرزند نمی باشد. بطور کلی کاربردهای چهار گانه این متد به شرح زیر می باشند:

* با زدن دکمه AddCommandدر فرم، در صورتی که فرم مربوط به یک رابطه باشد از طریق همین متد داده جدید به خصوصیت-رابطه جاری فرم افزوده می شود.
* در فرایند اتصال خصوصیت-رابطه که در همین بخش اشاره شد.
* در ایجاد داده پیش فرض جدید برای خصوصیت رابطه در فرم تک داده
* AddDataViewItemsToForm

### ایجاد داده پیش فرض برای فرم خصوصیت-رابطه تک داده:

همانگونه که پیشتر نیز اشاره شد در صورتی که فرم تک داده برای خصوصیت-رابطه ای نمایش داده شود و داده ای در رباطه موجود نباشد نیاز است تا یک داده بصورت پیش فرض در فرم بارگزاری گردد. به این منظور از متد CreateDefaultDataChildRelationship استفاده می شود (ChildRelationshipInfo.CreateDefaultDataChildRelationship: db3903adaa4d). در این متد ابتدا داده از طریق متد CreateAreaInitializerNewData ایجاد شده و سپس از طریق متد AddDataToChildRelationshipData به خصوصیت رابطه افزوده می گردد.

در اینجا تشریح متد SetBinding خصوصیت-رابطه ها و در پی آن متد نمایش داده در فرم مستقیم یا همان ShowDataInDataView به پایان می رسد. در ادامه متد نمایش داده ها در فرم غیر مستقیم بررسی می گردد.

### نمایش داده در فرم ورود اطلاعات خصوصیت-رابطه (ShowDataInChildRelationshipDataView):

هدف این متد نمایش داده در فرم ورود اطلاعات مرتبط یک خصوصیت-رابطه می باشد(ChildRelationshipInfo.ShowDataInChildRelationshipDataView: 78184401047a). برای نمایش داده در فرم رابطه از همان متد ShowDataInDataView مربوط به فرم های ورود اطلاعات که پیشتر تشریح گردید استفاده می شود. کاربردهای نمایش داده در فرم ورود اطلاعات خصوصیت-رابطه سه مورد می باشند :

* در متد افزودن داده به خصوصیت-رابطه که در بخش قبل به آن اشاره شد.
* در اتصال خصوصیت-رابطه (SetBinding) زمانی که فرم رابطه مستقیم بود.
* زمانی که در یک فرم غیر مستقیم دکمه نمایش فرم ورود اطلاعات زده می شود (ChildRelationshipInfo.DataViewRequested: e48af9744b57).

### نمایش داده در فرم غیر مستقیم خصوصیت رابطه (SetTempTextChildRelationshipData):

برای نمایش اطلاعات داده ها در فرم غیر مستقیم یک خصوصیت-رابطه از متد SetTempTextChildRelationshipData استفاده می شود (ChildRelationshipInfo.SetTempTextChildRelationshipData: 2da7f695849f). در این متد اطلاعات عمومی داده ها مانند تعداد و عنوان داده ها در نمای فرم غیر مستقیم نمایش داده می شوند. این متد دارای کاربردهای زیر می باشد :

* در افزودن داده از طریق متد AddDataToChildRelationshipData زمانی که فرم رابطه غیر مستقیم است.
* در زمان حذف داده از رابطه (ChildRelationshipInfo.RemoveRelatedData: 938077b01703)
* در زمان اتصال خصوصیت-رابطه ای که غیر مستقیم است و قبلاً داده هایش یکبار بارگزاری شده اند.

در بحث نمایش داده ها در فرم اصلی مستقیم به بحث اتصال رابطه و نهایتاً نمایش اطلاعات داده در خصوصیت-رابطه های فرم ورود اطلاعات و یا فرم غیر مستقیم رسیدیم. تنها بخش باقیمانده درباره نمایش داده ها متد نمایش داده ها در فرم اصلی غیر مستقیم می باشد. تنها حالتی که یک فرم کار با داده ممکن است اصلی و غیر مستقیم باشد همان حالت فرم مستقل انتخاب داده می باشد.

### نمایش داده در فرم اصلی غیر مستقیم (SetTempText):

برای نمایش اطلاعات داده ها در فرم اصلی از متد SetTempText استفاده می شود (BaseEditEntityArea.SetTempText: f03945399e2a). در این متد اطلاعات عمومی داده ها مانند تعداد و عنوان داده ها در نمای فرم غیر مستقیم نمایش داده می شوند. این متد دارای کاربردهای زیر می باشد :

* در زمان حذف داده از فرم اصلی (BaseEditEntityArea.RemoveData: e06d6fbdd00e)
* در زمان افزودن داده در فرم اصلی از طریق متد AddData

در ادامه به بررسی سایر متدهای مهم فرم های کار با داده می پردازیم.

### متد پاک کردن کلی داده ها از فرم اصلی (ClearData):

برای پاک کردن همه داده های فرم یا به عبارتی خالی کردن فرم اصلی از داده از این متد استفاده می شود(BaseEditEntityArea.ClearData: 2750b5bf6116). البته این متد در دل خود به ازای هر داده متد دیگری به نام RemoveData را فراخوانی می کند که عملیات اصلی پاک کردن هر داده به عهده آن است. کاربردهای متد حذف کلی داده یا همان ClearData موارد زیر می باشد:

* با زدن دکمه پاک کردن یا همان ClearCommand بروی فرم اصلی
* با زدن دکمه حذف داده (DeleteCommand) و پس از حذف موفق یکبار ClearData فراخوانی و فرم خالی می شود.
* با زدن دکمه ثبت داده (SaveCommand) و پس از ثبت موفق یکبار ClearData فراخوانی و فرم خالی می شود.
* در متد AddData در فرم اصلی، در صورت عدم نمایش موفق داده فرم توسط این متد خالی می شود
* AddDataViewItemsToForm
* با زدن دکمه پاک کردن در فرم غیر مستقیم (BaseEditEntityArea.TemporaryDisplayView\_TemporaryDisplayViewRequested: 95a95d63fa08)

### متد پاک کردن تک داده در فرم اصلی (RemoveData):

این متد عملیات پاک کردن داده ها را بصورت موردی انجام می دهد (BaseEditEntityArea.RemoveData: e06d6fbdd00e). برای پاک کردن داده عملیات زیر انجام می شود:

* داده از لیست داده ها حذف می شود.
* اگر فرم مستقیم بود
  + با فراخوانی متد ClearUIData ظاهر داده ها از واسط کاربری حذف می شود (رکوردهای جدول فرم چند داده ای)
  + اگر داده ای موجود نبود و فرم تک داده ای بود با فراخوانی CreateDefaultData یک داده پیش فرض ایجاد می گردد.
* اگر فرم غیر مستقیم بود، اطلاعات خلاصه داده ها بروزرسانی می گردد (SetTempText)
* نهایتاً ظاهر فرم و دکمه ها بروزرسانی می شود.

کاربرد های این متد موارد زیر می باشد:

* استفاده در متد پاک کردن کلی داده ها (ClearData) که بخش قبل اشاره شد
* استفاده در متد پاک کردن چند داده ای که در بخش بعد معرفی می شود(RemoveMultipleData).

### متد پاک کردن چند داده انتخابی در فرم اصلی (RemoveMultipleData):

کاربرد این متد زمانی است که بخواهیم چند داده انتخاب شده توسط کاربر را از فرم اصلی پاک کنیم(BaseEditEntityArea.RemoveMultipleData: 7b47193da8b7). تنها کاربرد آن در زدن دکمه RemoveCommand است که مختص فرم های چند داده ای است. به این ترتیب در صورتی که کاربر چند داده را انتخاب کرده باشد این متد پاک کردن چند داده ای فراخوانی می گردد که برای انجام این عملیات به ازای هر داده متد RemoveData را فراخوانی می کند.

### متد پاک کردن ظاهر فرم در واسط کاربری (ClearUIData):

به این متد یک داده معرفی می شود و تنها وظیفه اش پاک کردن ردیف معادل داده در واسط کاربری فرم های مستقیم چند داده می باشد (BaseEditEntityArea.ClearUIData: c3bf5a88417e). برای فرم های غیر مستقیم نیازی به اینکار نیست زیرا ظاهر فرم پس از تغییر داده ها هر بار با همان متد SetTempText بروز می شود. همچنین در فرمهای مستقیم تک داده نیز چون تنها یک داده در جریان است اطلاعات داده ها در واسط کاربری جایگزین می شوند و نیازی به پاک کردن اثر داده قبلی نمی باشد. کاربردهای این متد موارد زیر می باشند:

* پس از پاک کردن داده توسط متد RemoveData
* پس از پاک کردن داده مرتبط از خصوصیت-رابطه ظاهر داده نیز از واسط کاربری حذف می گردد.

### جستجو و انتخاب داده در فرم جستجو و نمایش داده :

یکی از عملیاتهای اصلی هر فرم کار با داده جستجو و انتخاب داده می باشد. درباره جزئیات فرم جستجو و نمایش در بخشهای قبل توضیحات کاملی ارائه شد. همانطور که اشاره شد کاربردهای فرم جستجو و نمایش داده در دکمه های جستجو چه در فرم های مستقیم و غیر مستقم و چه در فرم اصلی و یا فرمهای رابطه ها می باشد. در این بخش به رویدادهایی که داده های فرم جستجو و نمایش را به فرم کار با داده مرتبط می کند می پردازیم.

#### جستجو و نمایش داده در فرم نمایش داده :

پیشتر اشاره شد که در صورت تعیین عبارت جستجو رویداد SearchDataDefined فرم جستجو فراخوانی می شود(BaseEditEntityArea.SearchEntityArea\_SearchDataDefined: 3841062ffaca). این رویداد داده های نمایش را از متد GetDataViewItemsFromSearchResult دریافت و سپس از طریق متد AddData در فرم نمایش داده به آن فرم اضافه می کند. هدف متد GetDataViewItemsFromSearchResult جستجو داده های نمایشی طبق یک عبارت جستجوی مشخص می باشد (BaseEditEntityArea.GetDataViewItemsFromSearchResult: 7e4651a61540). در این متد با ساخت یک SearchViewRequest و فراخوانی SendSearchViewRequest (RequestRegistrationService.SendSearchViewRequest: 29b1b425fff4) داده های نمایشی از پایگاه داده هدف واکشی می شوند.

#### افزودن داده پس از انتخاب داده در فرم نمایش داده (DataSelected):

زمانی که داده های نمایشی جستجو شده و در فرم نمایش داده نیز لیست شدند کاربر می تواند با انتخاب آنها را به فرم کار با داده منتقل کند. به این منظور ابتدا رویداد انتخاب داده یا همان DataSelected از فرم نمایش فراخوانی می گردد (BaseEditEntityArea.ViewEntityArea\_DataSelected: a377695e6d06). در این متد ابتدا فرم جستجو و نمایش داده بسته می شود و سپس داده ها از طریق متد SelectData به فرم کار با داده مربوطه اضافه می گردند. این متد نیز در ادامه تشریح شده است.

### متد انتخاب داده در فرم کار با داده (SelectData):

هدف این متد انتخاب داده در فرم کار با داده بر اساس لیست داده ارسالی از نوع List<DP\_BaseData> می باشد(BaseEditEntityArea.SelectData: 1625660fbdfb). عملیات انتخاب داده تنها مختص فرم نمایش داده نمی باشد به این منظور به غیر از داده نمایشی ممکن است داده های پایه نیز به این متد ارسال شوند. البته در نهایت تمامی این داده ها می بایست به داده فرمی تبدیل و سپس به فرم افزوده شوند. کاربردهای این متد به شرح زیر می باشد:

* بعد از انتخاب داده در فرم نمایش و انتخاب داده توسط کاربر که پیشتر اشاره شد
* در مقداردهی مستقیم ستون داده فرمی زمانی که ستون کلید خارجی است و نیاز است تا داده مرتبط در فرم خصوصیت-رابطه مرتبط انتخاب شود (DP\_FormDataRepository.SetColumnValue: 5d7ba8eeba1a)
* در فرم جستجوی تعریف شده زمانی که فیلتر جستجو یک رابطه است و نیاز است تا قلم داده های پیش فرض و از قبل ذخیره شده در کنترل رابطه انتخاب شوند (EntityDefinedSearchArea.ShowPreDefinedSearch: cd004ddc85ff)

امّا عملکرد متد انتخاب داده چگونه است؟ در زیر مراحل انجام عملیات انتخاب داده شرح داده شده است:

* ابتدا عبارت جستجو با توجه به داده های ارسالی ساخته می شود.
* اگر فرم مستقیم باشد یا فرم غیر مستقیمی که فرم ورود اطلاعات آن باز است، داده های داینامیک با استفاده از عبارت جستجو از متد SendSearchEditRequest دریافت شده و لیست داده های فرمی مورد انتظار ساخته می شود.
* اگر فرم غیر مستقیمی باشد که فرم ورود اطلاعات آن باز نیست، لیست داده های نمایشی با استفاده از عبارت جستجو از متد SendSearchViewRequest دریافت شده و لیست داده های فرمی مورد انتظار ساخته می شود.
* سپس اگر فرم جاری فرم اصلی یا همان ریشه باشد، ابتدا تمام داده ها توسط متد ClearData پاک می شوند و سپس داده ها تک به تک توسط متد AddData به فرم افزوده می گردند.
* اگر فرم جاری از رابطه بوجود آمده باشد، ابتدا داده های مرتبط خصوصیت-رابطه مرتبط توسط متد RemoveRelatedData از رابطه حذف شده و اولین داده فرمی از لیست داده های فرمی از طریق متد AddDataToChildRelationshipData به خصوصیت-رابطه افزوده می گردد.

با اتمام مطالب مرتبط با جستجو و انتخاب داده به ادامه متدهای مهم فرم کار با داده می پردازیم.

### نمایش فرم ورود اطلاعات در فرمهای غیر مستقیم:

همانطور که پیشتر ذکر شد، فرمهای غیر مستقیم که از رابطه بوجود آمده اند قابلیتی دارند تا در صورت نیاز پنجره ورود یا اصلاح اطلاعات برای داده های آنها باز گردد. برای این عملیات از متد DataViewRequested استفاده می گردد (ChildRelationshipInfo.DataViewRequested: e48af9744b57). عملیات زیر در این متد به ترتیب انجام می گردد:

* در صورتی که فرم ورود اطلاعات جاری قبلاً تولید نشده بود، یکبار ظاهر فرم از طریق متد GenerateDataView تولید می گردد (BaseEditEntityArea.GenerateDataView: 4ba73472fca2)
* ظاهر داده ها به وسیله متد ClearUIData از واسط کاربری حذف می شود.
* اگر داده ای برای خصوصیت-رابطه موجود نباشد
  + اگر فرم تک داده باشد، یک داده پیش فرض از طریق متد CreateDefaultDataChildRelationship ایجاد می شود.
* اگر داده موجود باشد
  + اگر داده های فرمی کامل نباشند از طریق متد ConvertDataViewToFullData همه داده ها به داده فرمی کامل تبدیل می شوند
  + سپس تمامی داده ها توسط متد ShowDataInChildRelationshipDataView در فرم رابطه نمایش داده می شوند.
* تنظیمات ظاهری رابطه انجام می شود.
* فرم ورود اطلاعات نمایش داده می شود.

#### متد تبدیل داده نمایشی به فرمی (ConvertDataViewToFullData):

همانگونه که در بخش قبل اشاره شد وظیفه این متد تبدیل داده های فرمی نمایشی یک خصوصیت-رابطه به داده های فرمی کامل است (ChildRelationshipInfo.ConvertDataViewToFullData: 28ad76d5595c). از داده های ارسال شده به این متد یک عبارت جستجو ساخته می شود و سپس داده های کامل از طریق متد SendSearchEditRequest دریافت می شوند. سپس خصوصیات داده های کامل برای هر داده نمایش معادل جایگذاری می شود.

### متد پاک کردن کلی داده ها از رابطه (RemoveAllRelatedData):

برای حذف کل داده های یک خصوصیت-رابطه از متد RemoveAllRelatedDataاستفاده می شود (ChildRelationshipInfo.RemoveRelatedData: d5d649262e21). البته این متد برای انجام عملیات حذف به ازای هر داده متد دیگری به نام RemoveRelatedData را فراخوانی می کند که در بخش بعد به آن اشاره خواهد شد. کاربردهای پاک کردن کلی داده ها موارد زیر می باشد:

* با زدن دکمه ClearCommand در فرم زمانی که فرم مربوط به یک رابطه است
* با زدن دکمه پاک کردن در فرم غیر مستقیم رابطه
* با انتخاب داده برای رابطه از فرم نمایش و انتخاب داده، داده های موجود رابطه حذف می شوند.

### متد حذف چند داده انتخاب شده (RemoveRelatedMultipleData):

این متد چند داده انتخاب شده توسط کاربر را از رابطه حذف می کند(ChildRelationshipInfo.RemoveRelatedMultipleData: 3757aab2e26e). برای حذف هر داده متد RemoveRelatedData فراخوانی می گردد. مورد کاربرد آن در زمان استفاده از دکمه RemoveCommand در فرم های ورود اطلاعات چند داده ای می باشد.

### متد حذف داده از رابطه (RemoveRelatedData):

این متد داده ها را بصورت موردی از یک رابطه حذف می کند(ChildRelationshipInfo.RemoveRelatedData: 938077b01703). به علت وابستگی احتمالی داده ها عملیات حذف در این متد دارای فرایند خاصی می باشد که در زیر به آن اشاره شده است:

* ابتدا کنترل می شود که داده قابل حذف می باشد یا خیر. برای مثال داده ای که رابطه اش فقط خواندنی باشد قابل حذف نمی باشد. یا اینکه در صورتی که داده به علت وابستگی در صورت حذف می بایست به صورت کلی از پایگاه داده حذف شود، ابتدا شرایط حذف آن بررسی می شود و در صورت نیاز از کاربر تایید آن سوال پرسیده می شود. در صورت تایید کاربر عملیات حذف ادامه می یابد. برای کنترلهای ذکر شده از متد CheckRemoveData استفاده می شود که در ادامه به جزئیات آن پرداخته خواهد شد. اگر داده قابل حذف باشد ادامه مراحل انجام می شوند.
* داده ها از خصوصیت-رابطه حذف می شوند.
* مقادیر کلید خارجی در صورت نیاز توسط متد CheckRelationshipChanged مقداردهی می شوند.
* اگر رابطه غیر مستقیم باشد خلاصه داده ها توسط SetTempTextChildRelationshipData تنظیم می شود
* اگر فرم مستقیم باشد و یا فرم ورود اطلاعات فرم غیر مستقیم باز باشد
  + اطلاعات ظاهری داده توسط متد ClearUIData از فرم رابطه حذف می شود.
  + اگر داده ای وجود نداشته باشد و فرم تک داده باشد یک داده پیش فرض توسط متد CreateDefaultDataChildRelationship ایجاد می شود.
* تنظیمات ظاهری رابطه توسط SetControlUIDetails انجام می شود.

#### متد کنترل حذف پذیر بودن داده از رابطه (CheckRemoveData):

سسسسس

دسترسی حذف موجودیت نیز در صورتی که داده باید حذف شود بررسی شود

### متد مقداردهی ستونهای کلید خارجی هنگام تغییر داده های روابط (CheckRelationshipChanged):

زمانی که داده های یک رابطه تغییر می کنند طبیعتاً ستون های کلید خارجی تحت تاثیر تغییرات نیاز به مقداردهی مجدد دارند. برای کنترل این مقادیر از متد CheckRelationshipChanged استفاده می شود (ChildRelationshipInfo.CheckRelationshipChanged: e7c6d51dcb3d). کاربردهای این متد موارد زیر می باشد:

* به هنگام افزودن داده به رابطه در متد AddDataToChildRelationshipData
* به هنگام حذف داده از رابطه در متد RemoveRelatedData

روش کار این متد به شرح زیر است :

* اگر رابطه از اصلی به فرعی باشد و داده حذف شده باشد از متد خارج می شود زیرا داده کلید خارجی حذف شده و مقداردهی ستونها آن ضرورتی ندارد.
* داده طرف کلید اصلی و داده طرف کلید خارجی و به تبع آن ستونهای کلید اصلی و ستونهای کلید خارجی مشخص می شوند
* لیست مقادیر برای کلیدهای خارجی ساخته می شود و به ازای هر جفت ستون درگیر در رابطه
  + ستون کلید خارجی با مقدار ستون کلید اصلی معادل به لیست مقادیر اضافه می شود
  + ستون کلید خارجی با مقدار Null به لیست مقادیر اضافه می شود
* ستونهای کلید خارجی با استفاده از لیست مقادیر توسط متد SetColumnValue مقداردهی می شوند. درباره این متد و کاربردهایش در بخشهای بعدی توضیحاتی ارائه خواهد شد.

## نحوه اعمال دسترسی ها در فرم های کار با داده :

در این بخش نحوه اعمال دسترسی ها در فرم های ورود اطلاعات بررسی می شود. منظور از دسترسی ها حالتهای عدم دسترسی یا فقط خواندنی انواع اقلام درگیر در فرمهای ورود اطلاعات به مانند موجودیت ها، ستونها و روابط می باشد.

### دسترسیهای موجودیت :

در این بخش دسترسی هایی که روی موجودیتهای هر فرم کار با داده تاثیر گذار هستند بررسی می شود.

#### عدم دسترسی به موجودیت :

عدم دسترسی به موجودیت در فرم های کار با داده می تواند به دو علت باشد، یکی عدم فعال بودن کلی موجودیت و دیگری عدم دسترسی امنیتی کاربر به موجودیت. این دسترسی ها در دو سطح برای فرمهای کار با داده اعمال می گردند، سطح اول در همان متد GetEditEntityArea می باشد که سازنده فرم کار با داده برای یک موجودیت می باشد. در این متد در صورت عدم دسترسی به موجودیت فرم کار با داده تولید نخواهد شد. سطح دوم در ساخت موجودیت ورود اطلاعات برای فرمهای مستقیم چه فرم اصلی و چه فرم های روابط می باشد که در صورت عدم دسترسی در متد GetDataEntryEntity خروجی موجودیت ورود اطلاعات تولید نخواهد شود.

#### دسترسی فقط خواندنی موجودیت :

دسترسی فقط خواندنی موجودیت در مورد فرمهای ورود اطلاعات معنی دارد و بروی موجودیت ورود اطلاعات هر فرم موجود است. دلیل فقط خواندنی شدن موجودیت می تواند به دو علت باشد. یکی فقط خواندنی بودن کلی موجودیت و دیگری دسترسی فقط خواندنی کاربر به موجودیت. این دسترسی در فرم های ورود اطلاعات معنی پیدا می کند (چه فرم مستقیم و چه فرم ورود اطلاعات که از فرم غیر مستقیم باز شده است). بنابراین این دسترسی بروی موجودیت ورود اطلاعات هر فرم موجود است. در فرم های کار با داده این دسترسی در سه مورد تاثیرگذار می باشد :

* در متد تعیین دسترسی های دکمه های فرم (BaseEditEntityArea.DecideButtonsEnablity1: 195f736ab331) در صورتی که فرم ریشه باشد و در لیست داده ها مورد جدیدی وجود داشته باشد، دکمه ثبت داده ها (SaveCommand) غیر فعال می شود.
* در متد تعیین دسترسی های دکمه های فرم ورود اطلاعات ریشه (DecideButtonsEnablity1) دکمه افزودن داده (AddCommand) غیر فعال می شود.
* در متد تعیین دسترسی های دکمه های فرم غیر مستقیم رابطه (SetControlUIDetails) اگر داده موجود نباشد، دکمه باز شدن فرم ورود اطلاعات غیر فعال می شود.
* به هنگام ایجاد داده جدید در صورتی که داده پیش فرض فرم تک داده باشد، به علت فقط خواندنی بودن موجودیت، داده بلااستفاده می شود و خصوصیت IsUseLessBecauseNewAndReadonly آن مقداردهی می شود. حال اگر این داده در فرم ورود اطلاعات چه فرم ریشه و چه فرم ورود اطلاعات یک رابطه نمایش داده شده باشد کلاً بخش کنترلهای فرم غیر فعال می شود.

#### عدم دسترسی حذف داده :

عدم دسترسی حذف داده های موجودیت در دو حالت تاثیر گذار است، یکی برای فرمهای ورود اطلاعات ریشه زمانی که کاربر قصد حذف داده ای را داشته باشد و حالت دوم زمانی است که کاربر قصد حذف داده ای را از رابطه ای داشته باشد و این حذف به معنای حذف داده از پایگاه داده باشد. در این صورت نیز دسترسی حذف نیاز به کنترل شدن دارد. در ادامه این دو حالت بررسی می شوند.

* **عدم دسترسی حذف برای فرم ورود اطلاعات ریشه:** در این حالت این دسترسی توسط خصوصیت HasNotDeleteAccess بروی موجودیت ورود اطلاعات تعیین می شود. در صورت عدم دسترسی در متد تعیین دسترسی های دکمه های فرم دکمه حذف داده ها (DeleteCommand) غیر فعال می شود.
* این حالت دوم که اشاره شده اعمال شود

### دسترسی کنترلهای خصوصیات ساده :

در این بخش دسترسی هایی که روی خصوصیت-کنترل های ساده هر فرم کار با داده بررسی می شود. طبیعتاً این دسترسی ها تنها در فرم های ورود اطلاعات مطرح می باشند و نه فرمهای انتخاب داده.

#### عدم دسترسی به خصوصیت-کنترل :

عدم دسترسی به یک خصوصیت-کنترل در فرم های ورود اطلاعات می تواند به یکی از چهار علت باشد زیر باشد:

* عدم فعال بودن کلی ستون
* عدم قابلیت ورود اطلاعات ستون (DataEntryEnabled)
* یا عدم دسترسی امنیتی کاربر به ستون
* مخفی شدن خصوصیت-کنترل به علت وضعیت داینامیک اعمال شده بروی داده در فرم ورود اطلاعات

سه مورد اول اساساً باعث حذف کلی ستون در موجودیت ورود اطلاعات می شود و بنابراین ستون در فرم نمایش داده نمی شود. امّا مورد آخر ممکن است با توجه به تغییر مقادیر داده در فرمهای ورود اطلاعات بصورت داینامیک بروی خصوصیت-کنترلها اعمال شود و باعث مخفی شدن ستونی برای یک داده خاص شود. در این حالت وضعیت مخفی بودن یک خصوصیت-کنترل از خصوصیت IsHiddenOnState آن مشخص می شود و اعمال آن نیز در واسط کاربری از طریق همان متد تنظیمات ظاهری خصوصیت-کنترل می باشد به نام SetControlUIDetails می باشد(ChildSimpleContorlProperty.SetControlUIDetails: 2b095307e547). به این ترتیب و در صورت مثبت بودن این خصوصیت کنترل مورد نظر از واسط کاربری حذف می گردد.

#### تاثیرات دیگر عدم دسترسی به خصوصیت-کنترل:

عدم دسترسی به یک خصوصیت-کنترل بر اثر وضعیت (IsHiddenOnState) علاوه بر تاثیرات ظاهری، کاربردهای دیگری نیز دارد که در زیر به آنها اشاره شده است:

* در متد SetColumnValue به منظور مقداردهی ستون در صورت عدم دسترسی مقداردهی انجام نخواهد شد.
* در متد SetColumnValueRangeFromState به منظور تعیین لیست داده در صورت عدم دسترسی عملیات انجام نخواهد شد.
* در متد ResetColumnValueRangeFromState به منظور تعیین لیست داده در صورت عدم دسترسی عملیات انجام نخواهد شد.
* عملیات اتصال در متد SetBinding انجام نخواهد شد.
* در بروزرسانی باید در نظر گرفته شود و خصوصیتها حذف شوند

#### دسترسی فقط خواندنی خصوصیت-کنترل:

دسترسی فقط خواندنی خصوصیت-کنترلها در فرم ورود اطلاعات می تواند به دو دلیل اصلی باشد. دلیل اول دسترسی فقط خواندنی ستون در موجودیت ورود اطلاعات است که می تواند به علت فقط خواندنی بودن کلی ستون و یا دسترسی فقط خواندنی کاربر روی ستون باشد. دلیل اصلی دوم فقط خواندنی شدن خصوصیت-کنترل بر اساس وضعیت اعمال شده به روی داده خوانده شده از پایگاه داده می باشد. درباره وضعیتها و نحوه اعمال آنها در بخشهای بعدی توضیحاتی ارائه خواهد شد. فقط خواندنی بودن یک خصوصیت-کنترل از خصوصیت IsReadonly مشخص می گردد و نحوه اعمال آن در واسط کاربری در متد SetControlUIDetails می باشد (ChildSimpleContorlProperty.SetControlUIDetails: 2b095307e547). در این متد در صورت فقط خواندنی بودن خصوصیت-کنترل ضمن اعمال آن در واسط کاربری، پیغامی مرتبط بروی کنترل نمایش داده می شود.

#### تاثیرات دیگر دسترسی فقط خواندنی خصوصیت-کنترل:

دسترسی فقط خواندنی به یک خصوصیت-کنترل علاوه بر تاثیرات ظاهری، کاربردهای دیگری نیز دارد که در زیر به آنها اشاره شده است:

* در متد SetColumnValue به منظور مقداردهی ستون در صورت دسترسی فقط خواندنی مقداردهی انجام نخواهد شد.
* در متد SetColumnValueRangeFromState در صورت دسترسی فقط خواندنی عملیات انجام نخواهد شد.
* در متد ResetColumnValueRangeFromState در صورت دسترسی فقط خواندنی عملیات انجام نخواهد شد.
* در بروزرسانی باید در نظر گرفته شود و خصوصیتها حذف شوند

### دسترسی کنترلهای رابطه:

در این بخش دسترسی هایی که روی موجود در یک فرم ورود اطلاعات بررسی می گردد.

#### عدم دسترسی به رابطه :

عدم دسترسی به یک رابطه در فرم های ورود اطلاعات می تواند به یکی از چهار علت باشد زیر باشد:

* عدم فعال بودن کلی رابطه
* عدم قابلیت ورود اطلاعات رابطه (که روی کلید خارجی رابطه نگهداری می شود)
* یا عدم دسترسی امنیتی کاربر به رابطه (که روی کلید خارجی رابطه نگهداری می شود)
* به علت وضعیت داینامیک اعمال شده بروی داده های طرفین رابطه در فرم ورود اطلاعات

تفاوت سه مورد اول با مورد چهارم اینست که آنها به روی خود رابطه تعریف می شوند و در صورت عدم دسترسی بطور کل از موجودیت ورود اطلاعات حذف می شوند و بنابراین رابطه در فرم نمایش داده نمی شود. امّا مورد آخر به روی داده تعریف می شود و این داده ممکن است در فرم طرف اول رابطه باشد و یا در طرف فرم فرزند. اگر وضعیت عدم دسترسی رابطه برای داده در فرم ورود اطلاعات طرف اول باشد آنگاه طبیعتاً داده یک داده کامل و رابطه یک خصوصیت-رابطه خواهد بود و وضعیت رابطه از طریق ابزار مدیریت وضعیت ها بصورت داینامیک کنترل خواهد شد. یعنی ممکن است با تغییر شروط، وضعیت خصوصیت-رابطه بصورت لحظه ای مخفی شود و یا نمایش داده شود که بروی خصوصیت IsHiddenOnState خصوصیت رابطه تعیین می گردد. امّا اگر وضعیت بروی طرف دوم رابطه بود یعنی رابطه پدید آورنده یک فرم برای یک داده از آن فرم دچار عدم دسترسی می شود و داده می تواند کامل و یا نمایشی باشد. حال اگر داده اصیل باشد یعنی از ابتدا و در پایگاه داده در رابطه با داده پدر باشد، مخفی بودن رابطه پدر از ابتدای خواندن داده مشخص می شود زیرا وضعیت داده به هنگام خواندن داده از ابتدا و توسط خصوصیت ParentRelationshipIsHidenOnLoad کنترل شده است. در این حالت آن داده در فرم غیرفعال می شود و هیچ عملیاتی نیز بروی آن امکان پذیر نیست، یعنی اگر فرم تک داده باشد کل کنترل فرم و اگر فرم چند داده باشد آن ردیف داده غیر فعال خواهد شد (در متد SetControlUIDetails) . امّا اگر داده طرف دوم توسط کاربر در رابطه آمده باشد (یا داده جدید باشد یا در رابطه انتخاب شود)، آنگاه داده غیرفعال نخواهد شد تا امکان تغییر وضعیت داده همچنان در وابط کاربری موجود باشد امّا وضعیت داده قبل از ثبت پایش خواهد شد و در صورت عدم دسترسی به رابطه پدر از داده صرف نظر خواهد شد و پیغام مناسب نمایش داده می شود. در ادامه تاثیرات حالات ذکر شده بر برنامه دسته بندی شده اند:

##### تاثیرات عدم دسترسی به خصوصیت-رابطه:

همانطور که ذکر شد عدم دسترسی بروی یک خصوصیت رابطه از طریق خصوصیت IsHiddenOnState مشخص می شود و تاثیرات زیر را در یک فرم ورود اطلاعات دارد:

* مخفی شدن خصوصیت رابطه از طریق متد تنظیمات ظاهری خصوصیت-رابطه (ChildRelationshipInfo.SetControlUIDetails: a579903e12bd).
* در متد SetColumnValue در صورت مقداردهی ستونهای خارجی رابطه عملیات انجام نخواهد شد.
* عملیات اتصال در SetBinding انجام نخواهد شد.
* در دریافت داده های بروزرسانی و اعتبار سنجی ها

##### تاثیرات عدم دسترسی رابطه پدر برای داده طرف دوم اصیل:

* در همان متد SetControlUIDetails در خصوصیت رابطه مربوطه، داده غیر فعال می شود
  + اگر فرم تک داده باشد، کل فرم خصوصیت رابطه غیر فعال می شود
  + اگر فرم چند داده باشد و فرم ورود اطلاعات باز باشد، تنها ردیف متناظر داده غیر فعال می شود.
* در حذف داده توسط کاربر یا جایگذاری به هنگام انتخاب کنترل رابطه انجام می شود.

##### تاثیرات عدم دسترسی رابطه پدر پس از لود:

* اعتبارسنجی و کنترل به هنگام بروزرسانی.

#### دسترسی فقط خواندنی رابطه:

دسترسی فقط خواندنی خصوصیت-رابطه در فرم ورود اطلاعات می تواند به دو دلیل اصلی باشد. دلیل اول دسترسی فقط خواندنی رابطه در موجودیت ورود اطلاعات است که می تواند به علت فقط خواندنی بودن کلی رابطه و یا دسترسی فقط خواندنی کاربر روی رابطه باشد. دلیل اصلی دوم فقط خواندنی شدن رابطه بر اساس وضعیتهای اعمال شده بروی داده های خوانده شده از پایگاه داده تعریف می شود و این داده ممکن است در فرم طرف اول رابطه باشد و یا در طرف فرم فرزند. از آنجایی به داده های خوانده شده از پایگاه داده اشاره شده که اساساً دسترسی فقط خواندنی تنها برای داده های موجود در پایگاه داده معنی دارد. اگر داده در طرف اول رابطه باشد طبیعتاً داده کامل خواهد بود و رابطه به صورت یک خصوصیت رابطه برای این داده وجود خواهد داشت. در این حالت فقط خواندنی بودن خصوصیت-رابطه همراه با خواندن داده پدر دریافت شده و از طریق خصوصیت IsReadonlyOfState بروی خصوصیت-رابطه مشخص خواهد بود. امّا اگر وضعیت بروی طرف دوم رابطه بود یعنی رابطه پدید آورنده داده در فرم دچار شرط فقط خواندنی می شود و این وضعیت از همان ابتدای خواندن داده اصیل برای رابطه بروی خصوصیت ParentRelationshipIsReadonlyOnLoad مشخص می شود. در ادامه تاثیرات حالات ذکر شده بر برنامه دسته بندی شده اند:

##### تاثیرات دسترسی فقط خواندنی به خصوصیت رابطه :

فقط خواندنی بودن خصوصیت رابطه از خصوصیت IsReadonly آن مشخص می شود که همانطور که اشاره شد تلفیقی از دسترسی فقط خواندنی به ستون و همچنین وضعیت فقط خواندنی خصوصیت رابطه می باشد.

* اعمال ظاهری بروی خصوصیت رابطه از طریق متد تنظیمات ظاهری خصوصیت-رابطه (ChildRelationshipInfo.SetControlUIDetails: a579903e12bd) که مراحل زیر به جزء انجام می شود:
  + پیغامی مناسب درباره فقط خواندنی شدن رابطه نمایش داده می شود.
  + دکمه پاک کردن داده ها در فرم رابطه چه در فرم ورود اطلاعات و چه فرم غیر مستقیم غیرفعال می شود.
  + دکمه جستجوی داده در فرم رابطه چه در فرم ورود اطلاعات و چه فرم غیر مستقیم غیرفعال می شود. همچنین دکمه های جستجوی میانبر در فرم غیر مستقیم.
  + دکمه افزودن داده در فرم ورود اطلاعات غیر فعال می شود.
  + دکمه حذف داده در فرم ورود اطلاعات غیر فعال می شود.
  + اگر داده موجود نباشد، دکمه فرم ورود اطلاعات در فرم غیر مستقیم غیر فعال می شود.
* در متد SetColumnValue به منظور مقداردهی ستون های کلید خارجی رابطه و انتخاب داده ها در صورت دسترسی فقط خواندنی مقداردهی انجام نخواهد شد.
* به هنگام ایجاد داده جدید در صورتی که داده پیش فرض فرم تک داده باشد، به علت فقط خواندنی بودن رابطه، داده بلااستفاده می شود و خصوصیت IsUseLessBecauseNewAndReadonly آن مقداردهی می شود. حال اگر این داده در فرم ورود اطلاعات رابطه نمایش داده شود کلاً بخش کنترلهای فرم غیر فعال می شود.
* در بروزرسانی باید در نظر گرفته شود و خصوصیتها حذف شوند

##### تاثیرات دسترسی فقط خواندنی به رابطه پدر برای داده طرف دوم:

* اگر خصوصیت ParentRelationshipIsReadonlyOnLoad داده مثبت باشد در صورتی که داده اصیل باشد به هنگام جایگزینی یا حذف داده به هر دلیل (حذف مستقیم و یا انتخاب و جایگزینی داده دیگر) با نمایش پیغامی عملیات انجام نخواهد شد. همچنین اگر داده توسط کاربر انتخاب شده باشد از جایگزینی داده جلوگیری خواهد شد.

## وضعیتهای داده و نحوه اعمال آنها :

یکی از قابلیتهای سامانه ورود اطلاعات، امکان تعریف وضعیتهایی بروی موجودیت ها در صورت محقق شدن برخی شروط می باشد. نهایتاً هر وضعیت منجر به انجام عملی روی خصوصیات داده می گردد. هر وضعیت در قالب موجودیتی از نوع EntityStateDTO نگهداری می گردد که ساختار آن در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه وضعیت |
| TableDrivedEntityID | int | شناسه موجودیت مرتبط |
| Title | string | عنوان وضعیت |
| ActionActivities | List<UIActionActivityDTO> | اقدامات وضعیت |
| RelationshipTailID | int | شناسه رشته رابطه ستون هدف |
| RelationshipTail | EntityRelationshipTailDTO | رشته رابطه ستون هدف |
| ColumnID | int | شناسه ستون هدف |
| Column | ColumnDTO | ستون هدف |
| FormulaID | int | شناسه فرمول |
| Formula | FormulaDTO | فرمول |
| Values | List<EntityStateValueDTO> | لیست مقادیر مقایسه |
| EntityStateOperator | InORNotIn | عملگر مقایسه مقادیر |
| SecuritySubjects | List<int> | موضوعات امنیتی |
| SecuritySubjectInORNotIn | InORNotIn | عملگر موضوعات امنیتی |

### شرط هر وضعیت :

خصوصیات مربوط به شرط وضعیت بخش مهمی از تعریف وضعیت می باشد. تعیین یک شرط یا از طریق مقدار یک ستون مرتبط با داده (ستون خود داده و یا از رشته رابطه ای مرتبط) و یا تعیین مقدار یک فرمول بروی داده می باشد. تعیین ستون و رشته رابطه احتمالی در خصوصیات ColumnID و RelationshipTailID و فرمول نیز در خصوصیت FormulaID می باشد. مقدار بدست آمده از ستون یا فرمول با لیست مقادیر (خصوصیت Values) با شرط شامل باشد و یا شامل نباشد (InORNotIn) مقایسه می شود. البته هر مقدار مقایسه از نوع EntityStateValueDTO است که می تواند دارای یک مقدار مستقیم باشد و یا یک مقدار رزرو شده از نوع SecurityReservedValue باشد. مقادیر رزور شده به مانند شناسه کاربر جاری و یا تاریخ روز متغیرهای میانبری هستند که مقادیر آنها بصورت داینامیک با توجه به محیط برنامه در حال اجرا تولید می گردند. به هر شرط زیر شرط دیگری هم می توان اضافه نمود که نقشهای کاربر جاری را نیز در شرط لحاظ نمود. این زیر شرط در متغیر SecuritySubjects قابل تعریف است. به این ترتیب در صورت وجود نقشهایی در این خصوصیت، علاوه بر مقایسه مقادیر شرط، نقش کاربر جاری نیز می بایست در لیست نقشهای SecuritySubjects صدق کند (در لیست باشند و یا نباشد طبق عملگر SecuritySubjectInORNotIn).

### آثار هر وضعیت :

اما در صورت صحت شرط داده آن وضعیت بروی داده برقرار می گردد و تغییراتی بروی داده اعمال می گردد. نوع و جزئیات این تغییرات در خصوصیت ActionActivities نگهداری می شوند. انواع این تغییرات موارد زیر می باشد:

* فقط خواندنی سازی کل خصوصیات داده
* عمیلیات بروی خصوصیات و روابط شامل
  + فقط خواندنی سازی ستونها و روابط مشخص داده
  + مخفی سازی ستونها و روابط مشخص داده
* تعیین مقدار یک خصوصیت
* تعیین بازه داده یک خصوصیت

به هر تغییر ممکن یک اثر وضعیت اطلاق می کنیم که هر کدام از نوع UIActionActivityDTO می باشد و دارای ساختار زیر است :

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه اثر |
| Type | Enum\_ActionActivityType | نوع اثر |
| Title | string | عنوان اثر |
| UIEnablityDetails | List<UIEnablityDetailsDTO> | جزئیات عملیات بروی کنترلها |
| UIColumnValue | List<UIColumnValueDTO> | لیست مقادیر ستون |
| UIColumnValueRange | UIColumnValueRangeDTO | بازه داده ستون |

خصوصیت Type نشان دهنده نوع اثر است که انواع آن پیشتر ذکر شد. در ادامه به هر نوع اثر بصورت اختصاصی پرداخته می شود:

#### اثر بروی خصوصیات و روابط:

در صورتی که قصد انجام مانند مخفی سازی و یا فقط خواندنی سازی لیستی از ستونها و یا روابط را داشته باشیم از این نوع اثر وضعیت استفاده می کنیم. در این صورت خصوصیت Type برابر با UIEnablity قرار می گیرد و لیست این تغییرات در خصوصیت UIEnablityDetails قرار می گیرد که ساختار آن در زیر اشاره شده.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه مورد |
| ColumnID | int | شناسه ستون |
| RelationshipID | int | عنوان اثر |
| Hidden | Bool? | عملیات بروی کنترلها |
| Readonly | Bool? | لیست مقادیر ستون |

هر مورد یا از نوع مخفی سازی است یا از نوع فقط خواندنی سازی که یا روی یک خصوصیت اعمال می گردد یا روی یک رابطه از داده.

#### فقط خواندنی سازی کل خصوصیات داده:

در صورتی که نیاز باشد تا بصورت مستقیم کل داده فقط خواندنی شود از این نوع اثر وضعیت استفاده می کنیم. در این صورت خصوصیت Type برابر با EntityReadonly قرار می گیرد. فقط خواندنی شدن داده به معنی فقط خواندنی شدن تمام خصوصیات داده شامل خصوصیات ساده و همچنین روابط فرعی به اصلی داده می باشد.

#### تعیین مقدار یک خصوصیت:

امکان تعیین مقدار یک خصوصیت نیز یکی از آثار ممکن وضعیتها می باشد. در این صورت خصوصیت Type برابر با ColumnValue می شود و از خصوصیت UIColumnValue برای نگهداری جزئیات این آثار استفاده می گردد که ساختار آن در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه مورد |
| ColumnID | int | شناسه ستون هدف |
| ExactValue | object | مقدار مستقیم |
| FormulaID | int | شناسه فرمول |
| ReservedValue | SecurityReservedValue | مقدار رزرو شده |
| EvenHasValue | bool | حتی اگر مقدار دارد |
| EvenIsNotNew | bool | حتی اگر داده جدید نیست |

مقدار خصوصیت ممکن از یکی از این سه منبع دریافت شود :

* مقداردهی مستقیم که در خصوصیت ExactValue مشخص می شود.
* از مقدار نتیجه یک فرمول که در خصوصیت FormulaID مشخص می شود.
* یک مقدار رزور شده که از خصوصیت ReservedValue تعیین می گردد.

خصوصیت EvenHasValue تعیین کننده این شرط است که آیا مقداردهی در صورتی انجام شود که خصوصیت مقدارش خالیست یا خیر و خصوصیت EvenIsNotNew تعیین این شرط است که آیا مقداردهی تنها برای داده های جدید انجام شود و یا حتی داده موجود و خوانده شده از پایگاه داده.

#### تعیین بازه داده برای یک خصوصیت :

برای این نوع اثر وضعیت مقدار خصوصیت Type برابر با مقدار ColumnValueRange قرار می گیرد. هدف آن نیز تغیر بازه مقادیر یک خصوصیت بر اساس مقدار خصوصیت دیگر می باشد. جزئیات آن در خصوصیت UIColumnValueRange نگهداری می گردد که ساختار آن در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه مورد |
| ColumnID | int | شناسه ستون هدف |
| FilterValueRelationshipTailID | شناسه | شناسه رشته رابطه ستون مقدار فیلتر |
| FilterValueRelationshipTail | int | شناسه رشته رابطه ستون مقدار فیلتر |
| FilterValueColumnID | SecurityReservedValue | شناسه ستون مقدار فیلتر |

برای این نوع اثر وضعیت در مدل تعریفی صورت نمی پذیرد بلکه در فرم ورود اطلاعات از روی فیلترهای تعریف شده برای ستونهای دارای بازه مقادیر بصورت خودکار وضعیت مرتبط با آن فیلتر و آثار وضعیت مرتبط ساخته می شود. در واقع شناسه ستون همان ستونی است که دارای بازه مقادیر می باشد و FilterValueColumnID نیز شناسه خصوصیتی است که فیلتر بازه مقادیر باید طبق مقدار آن خصوصیت انجام گیرد. حال این خصوصیت مقدار فیلتر می تواند متعلق به همان داده باشد و یا از یک رشته رابطه بدست بیاید. در بخشهای بعدی که به نحوه اعمال وضعیتهای می پردازیم در این مورد توضیحات بیشتری ارائه خواهد شد.

### تعریف وضعیتها در مدل فراداده :

برای تعریف وضعیتهای یک موجودیت از فرم مخصوصی در برنامه مدیریت فراداده استفاده می شود(frmEntityStates: 1082873315d1). در این فرم عنوان وضعیت و شرط آن طبق ساختار اشاره شده تعریف می شوند. همچنین جدولی وجود دارد که لیست آثار وضعیت معرفی گردند. این آثار یا انتخاب می شوند و یا با استفاده از میانبر به فرم دیگری ارجاع می شود که مخصوص تعریف آثار طبق ساختار اشاره شده است(frmUIActionActivity: 6c1a76026666).

### کاربرد وضعیتها در برنامه :

مهمترین بخش بررسی وضعیتها نحوه بکارگیری آنها در برنامه ورود اطلاعات می باشد. اساساً فلسفه وضعیتها اعمال تغییرانی روی داده و خصوصیاتش بنابر شروطی است، بطوری که در واسط کاربری آن وضعیت توسط کاربر قابل مشاهده باشد. به عبارت دیگر هدف از وضعیتها تصمیم گیری و تغییر داده در لایه های پشت واسط کاربری نمی باشد و بیشتر کنترلهای بیزینسی بروی داده ها مد نظر است. آثار وضعیتها به دو دسته تقسیم می شوند، دسته اول آثاری که از ابتدای لود شدن داده در واسط کاربری بروی داده سنجش و اعمال شده اند و دسته دوم آثاری که بصورت داینامیک در فرم ورود اطلاعات داده سنجش و اعمال می شوند. چه در واسط کاربری و چه هنگام واکشی داده ها، لیست وضعیتها به منظور اعمال بروی داده ها از متد GetEntityStatesForApply دریافت می شود(BizEntityState.GetEntityStatesForApply: 829758bde96e). در این متد از طریق پارامتری به نام Enum\_ApplyState نوع وضعیتهای قابل اعمال مشخص می شود که در ادامه به تشریح انواع وضعیتها بیشتر پرداخته می شود.

### آثار وضعیت همراه داده، انواع و کاربرد:

برخی از آثار وضعیتها تنها بروی داده های موجود اعمال می شوند و قبل از واکشی داده ها به منظور نمایش در واسط کاربری و در صورت برقراری شرط وضعیت بروی داده سنجش و اعمال می شوند. این آثار در فرمهای ورود اطلاعات تا زمان ثبت همراه داده می مانند و تغییری در آنها ایجاد نمی شود. سنجش وضعیتهای مرتبط این آثار و نهایتاً اعمال آنها بروی داده همگی بعد از واکشی داده ها و قبل از ارسال به فرمهای کار با داده در متد DoDataBeforeLoadActionActivities انجام می شود(BizEntityState. DoDataBeforeLoadActionActivities: 84133990d0c1). این آثار شامل فقط خواندنی نمودن ستونها و خصوصیات داده و یا عدم دسترسی به رابطه پدر داده می باشد. برخی از این آثار بروی هر دو نوع داده کامل و نمایشی و برخی تنها بروی داده کامل اجرا می شوند. در نتیجه بسته به اینکه داده کامل است و یا نمایشی وضعیتهایی از متد GetEntityStatesForApply دریافت می شوند که اولاً دارای آثاری مناسب زمان واکشی داده و همچنین مناسب نوع داده بر حسب نمایشی بودن و یا کامل بودن داده باشند. در ادامه به بررسی بیشتر این آثار و نوع داده پرداخته می شود.

#### آثار قابل اعمال بروی داده نمایشی و کامل:

آثاری که بروی هر دو داده کامل و نمایشی قابل اعمال است آنهایی است که با رابطه ریشه داده کار دارد. حال ممکن است عدم دسترسی به رابطه و یا فقط خواندنی نمودن رابطه باشد. در مورد عدم دسترسی به رابطه ریشه نیز این حالت مستقیماً خود داده و در خصوصیتی به نام ParentRelationshipIsHidenOnLoad تعیین می گردد. در این حالت طبق توضیحات عدم دسترسی به رابطه ریشه در بخشهای قبل عمل می شود. یعنی اگر داده اصیل باشد آن داده در فرم غیرفعال می شود و هیچ عملیاتی نیز بروی آن امکان پذیر نیست، یعنی اگر فرم تک داده باشد کل کنترل فرم و اگر فرم چند داده باشد آن ردیف داده غیر فعال خواهد شد(در متد SetControlUIDetails). امّا اگر داده طرف دوم توسط کاربر در رابطه آمده باشد (یا داده جدید باشد یا در رابطه انتخاب شود)، آنگاه داده غیرفعال نخواهد شد تا امکان تغییر وضعیت داده همچنان در وابط کاربری موجود باشد امّا وضعیت داده قبل از ثبت پایش خواهد شد و در صورت عدم دسترسی به رابطه پدر از داده صرف نظر خواهد شد و پیغام مناسب نمایش داده می شود. فقط خواندنی نمودن رابطه ریشه نیز برای داده بروی خود داده و در خصوصیتی به نام ParentRelationshipIsReadonlyOnLoad تعیین می گردد. در صورتی که داده اصیل باشد به هر حال می بایست داده در رابطه فرزند نمایش داده شود، در این حالت همانطور که در بخش تاثیر دسترسی فقط خواندنی به رابطه پدر پیشتر اشاره شد در صورت اقدام به حذف یا جایگزینی داده با نمایش پیغام فقط خواندنی بودن رابطه از تغییر رابطه داده جلوگیری می شود. اگر داده اصیل نباشد و توسط کاربر انتخاب شود ضمن نمایشی پیغام فقط خواندنی بودن رابطه از انتخاب داده جلوگیری می شود(BaseEditEntityArea.SelectData: 1625660fbdfb).

#### آثار قابل اعمال صرفاً بروی داده کامل:

آثاری که صرفاً بروی داده کامل قابل اجرا هستند، فقط خواندنی نمودن خصوصیات و روابط فرزند داده می باشند. در این حالت طبیعتاً داده کامل قرار است واکشی شده و در یک فرم ورود اطلاعات نمایش داده شود. بنابراین لیست خصوصیات داده مهیا می باشد و فقط خواندنی بودن ستونهای ساده بروی همان خصوصیت معادل و به عنوان OnLoadIsReadonlyOfState مشخص می شود. اما خصوصیت-روابط داده یا همان ChildRelationshipDatas هنوز دارای مقدار نمی باشند زیرا همانطور که پیشتر اشاره شد این خصوصیت در فرمها بصورت درختی و در زمان نمایش داده ساخته می شود. بنابراین فقط خواندنی شدن خصوصیت-روابط موقتاً بروی خصوصیتی به نام ChildReadonlyRelationships بروی داده مشخص می شود. سپس در فرم ورود اطلاعات بعد از مقداردهی خصوصیت ChildRelationshipDatas برای روابط داده در فرمهای ورود اطلاعات فقط خواندنی بودن آن از همان خصوصیت ChildReadonlyRelationships تعیین می گردد.

### آثار وضعیت داینامیک داده در فرمهای ورود اطلاعات:

همانطور که مطرح شد برخی آثار وضعیت در واسط کاربری بروی داده اعمال می شوند و با تغییر داده ها و شروط ممکن است وضعیت آنها و نهایتاً آثار وضعیت برقرار و یا رفع شود که لیست آنها در زیر اشاره شده است:

* تعیین مقدار خصوصیت ها
* تعیین بازه مقادیر یک ستون
* عدم دسترسی به یک ستون و یا رابطه فرزند

تمامی این آثار زمانی که داده در حالت نمایش در فرم ورود اطلاعات است معنی پیدا می کند غیر از یک حالت استثنا که هم بروی داده در فرم ورود اطلاعات و هم داده در فرم غیر مستقیم اعمال می شود. آن حالت استثنا نیز در مورد عدم دسترسی به رابطه داده در صورتی که رابطه ریشه فرم فرزند به پدر باشد و تنها اگر داده غیر اصیل باشد(در صورت اصیل بودن داده به همان روش اثر همراه با داده که در بخش قبل توضیح داده شد با آن برخورد می شود).

#### سنجش وضعیتهای داینامیک و اعمال آنها:

وظیفه اصلی ابزار مدیریت وضعیتها پایش داده های موجود در فرم و سنجش و اعمال وضیعیتهای آنها می باشد. نکته مهم اینست که بنابر توضیحی که در بخش قبل اشاره شد، وضعیتها نیز بر حسب آثارشان به دو دسته تقسیم می شوند، یکی وضعیتهای مورد اعمال بروی داده چه داده کامل باشد و چه داده نمایشی و وضعیتهای مختص یک داده کامل (در یک فرم ورود اطلاعات). بنابراین عملیات سنجش و اعمال وضعیتها بر حسب اینکه داده ممکن است از چه نوعی و در چه وضعیتی باشد متفاوت می باشند. لیست وضعیتهای کلی قابل اجرا در واسط کاربری توسط همان متد GetEntityStatesForApply دریافت می شود امّا اینکه آیا یک وضعیت برای یک داده قابل سنجش است یا خیر از طبق قانون زیر تعیین می شود:

* اگر وضعیت دارای اثر قابل اجرا بروی داده نمایشی باشد (یعنی داده اصیل نباشد و اثر عدم دسترسی به ریشه داشته باشد)، آنگاه اگر داده در وضعیت نمایش در فرم غیر مستقیم و یا نمایش در فرم ورود اطلاعات باشد وضعیت قابل سنجش است.
* اگر وضعیت دارای اثر قابل اجرا بروی داده نمایشی نباشد، آنگاه تنها اگر داده در حالت نمایش در فرم ورود اطلاعات باشد وضعیت قابل سنجش است. بصورت کلی عملیات سنجش و اعمال داده ها در متد CheckAndImposeEntityStates انجام می گردد (UIActionActivityManager.CheckAndImposeEntityStates: 5a9f918cdb55).

روند سنجش وضعیتها و اعمال آنها به شرح زیر است:

* ابتدا تمامی آثار قابل برگشت تمامی وضعیتهای قابل سنجش بروی داده غیر فعال می شوند(UIActionActivityManager.ResetActionActivities: 42b13c5915e5). یعنی اگر داده در حالت نمایشی باشد شرط عدم دسترسی به رابطه پدر حذف می شود. اگر داده کامل و در حالت نمایش باشد علاوه بر شرط ذکر شده برای داده نمایشی، تمامی خصوصیات و روابط فرزند داده نیز از حالت عدم دسترسی خارج می شوند. هدف از این کار عدم تداخل فاز بررسی شرایط وضعیت با آثار وضعیت ها می باشد.
* سپس به ازای تمامی وضعیتهای مناسب داده، شرط برقراری وضعیت کنترل می شوند (UIActionActivityManager.CheckEntityState: 79bffbfb2ad2). در صورت برقراری شرط وضعیت بروی داده اجرایی می گردد (UIActionActivityManager.DoStateActionActivity: 9d894434cc60). در ادامه به نحوه اعمال انواع آثار وضعیتها بروی داده ها در واسط کاربری می پردازیم.

#### اعمال آثار داینامیک وضعیت بروی داده :

همانطور که اشاره شد برای اعمال یک وضعیت بروی داده از متد DoStateActionActivity استفاده می گردد. روند کار به شرح زیر است:

* به ازای هر آیتم اثر وضعیت روال زیر انجام می گردد
  + اگر اثر در لیست آثار وضعیت در حال اجرای داده باشد از اجرای آن صرف نظر می شود. در غیر اینصورت به لیست آثار در حال اجرای داده افزوده می گردد. کاربرد این لیست اینست که از تداخل آثار وضعیتها و تکرار آنها در یک حلقه جلوگیری می کند.
  + اگر نوع اثر از نوع عملیات عدم دسترسی بروی خصوصیات و روابط بود، به ازای تمامی موارد :
    - اگر مورد عدم دسترسی به رابطه ریشه بود، این کار از طریق مقدار دهی خصوصیت ParentRelationshipIsHidenInUI بروی خود داده انجام می شود. سپس در متد SetControlUIDetails برای این داده ها اقدام مناسب در جهت یادآوری عدم دسترسی به داده و رابطه پدرش انجام می شود.
    - اگر مورد دیگری بود، یعنی مخفی سازی ستونها و روابط فرزند داده، متد ChildItemVisiblity فراخوانی می گردد(UIActionActivityManager.ChildItemVisiblity: 438a981b8c49). در این متد برای خصوصیت-کنترلهای داده متد AddHiddenState فراخوانی می گردد که باعث مخفی شدن آن خصوصیت-کنترل می گردد(ChildSimpleContorlProperty.AddHiddenState: a301af124dd7) . برای خصوصیت-رابطه ها نیز به همین ترتیب متد AddHiddenState مخصوص آنها فراخوانی و خصوصیت-رابطه مخفی می گردد (ChildRelationshipInfo.AddHiddenState: c77deede58e4)
  + اگر نوع اثر مقداردهی یک ستون باشد، لیست ستونها و مقادیر مربوطه (خصوصیت UIColumnValue) به متد SetColumnValue داده ارسال می گردد. این متد وظیفه مقداردهی ستوهای یک رابطه را دارد (DP\_FormDataRepository.SetColumnValue: 5d7ba8eeba1a). در بخشهای بعدی به جزئیات این متد پرداخته خواهد شد.
  + اگر نوع اثر فیلتر نمودن بازه مقادیر یک ستون باشد، با استفاده از خصوصیت UIColumnValueRange در اثر وضعیت ابتدا لیست بازه مقادیر فیلتر شده بدست می آید (UIActionActivityManager.GetFilteredRange: 339cd0b07965) و سپس این لیست به متد SetColumnValueRangeFromState خصوصیت-رابطه هدف به منظور مقداردهی ارسال می شود (ChildSimpleContorlProperty.SetColumnValueRangeFromState: a9112d396ebd)
  + با اتمام روال اجرای اثر وضعیت، آن مورد از لیست آثار وضعیتها در حال اجرا حذف می گردد.
* در صورتی که فرم تک داده باشد و ستون یا رابطه ای از فرم مخفی شده باشد ظاهر فرم یکبار به منظور چینش مناسب و حذف کانتینرهای خالی کنترل می گردد (EditEntityAreaOneData.CheckContainersVisiblity: 29dc4e65ce6f)

در این بخش عملیات سنجش وضعیتها و اعمال آنها تشریح گردید. امّا این عملیات در چه زمانی صورت می پذیرد؟ در سه حالت داده ها در فرم های ورود اطلاعات سنجش وضعیت می شوند. ابتدا در صورت نمایش داده در فرم ورود اطلاعات، سپس در صورت تغییر مقادیر شروط وضعیت و نهایتاً هنگام ثبت و تایید فرم. در ادامه به هر سه حالت پرداخته می شود:

##### کنترل وضعیتهای داینامیک به هنگام نمایش داده در فرم:

هر زمانی که داده ای در یک فرم کار با داده نمایش داده می شود رویدادی نیز در ابزار کمکی وضعیتها به نام DataLoaded فراخوانی می گردد(UIActionActivityManager.DataLoaded: 37a441f938c0). به طور دقیق در چهار مورد این رویداد فراخوانی می گردد که در زیر به آنها اشاره شده است:

* در فرم ریشه زمانی که داده توسط متد AddData به فرم افزوده می شود.
* در فرمهای رابطه و در خصوصیت-رابطه مربوطه زمانی که داده توسط متد AddDataToChildRelationshipData به رابطه افزوده شده و نمایش داده می شود.
* در فرمهای رابطه و در خصوصیت-رابطه مربوطه زمانی که در متد SetBinding داده از قبل لود شده رابطه نمایش داده می شود.
* در فرمهای رابطه و در خصوصیت-رابطه مربوطه زمانی که داده از حالت نمایشی به داده کامل به منظور نمایش در فرم ورود اطلاعات تبدیل می شود.

در رویداد DataLoaded دو مورد انجام می شود، ابتدا عملیات سنجش و اعمال وضعیتها روی داده انجام می گردد و سپس در صورتی که داده برای اولین بار در فرم نمایش داده شده باشد عملیات آماده سازی ردیابی تغییرات شروط وضعیت بروی داده انجام می شود، به این ترتیب در صورتی که مقادیر خصوصیات و روابطی که بطور مستقیم در شرط وضعیت دخیل هستند برای داده تغییر نمود یکبار دیگر عملیات سنجش و اعمال وضعیت برای داده انجام می گیرد. در ادامه به این حالت بیشتر پرداخته می شود.

##### کنترل وضعیتها داینامیک به هنگام تغییر مقادیر خصوصیات و روابط مرتبط:

همانطور که ذکر شد بعد از نمایش هر داده در فرم ورود اطلاعات این رویداد توسط DataToShowInDataview به ابزار مدیریت وضعیتها اطلاع داده می شود. سپس علاوه بر اعمال وضعیتها ردیابهایی به داده ها متصل می شوند تا در صورت تغییر مقادیر شروط بصورت خودکار وضعیت مجدداً محاسبه و اعمال گردد. عملیات ایجاد این ردیابها در متد SetDataItemChangeMonitors آغاز میگردد (UIActionActivityManager.SetDataItemChangeMonitors: 96259224044f). در واقع هدف این متد آماده سازی داده برای پایش برخی خصوصیات و روابطش می باشد. در این متد روال زیر طی می شود:

* ابتدا چک می شود در صورتی که قبلاً این متد بروی داده انجام شده بود از انجام مجدد آن جلوگیری می شود.
* سپس لیستی خالی از رشته روابط و ستون های مورد پایش ساخته می شود. سپس،
  + اگر در شرط وضعیت از فرمول استفاده شده بود، به ازای آیتمهای درگیر در فرمول، رشته رابطه ها و ستون های مربوط به لیست مورد پایش افزوده می شوند.
  + اگر در شرط از مقدار یک ستون استفاده شده بود، آن ستون و رشته رابطه مربوطه اش به لیست مورد پایش افزوده می شود.
  + به ازای هر مورد موجود در لیست پایش متد AddChangeMonitorIfNotExists بروی داده فراخوانی می گردد(DP\_FormDataRepository.AddChangeMonitorIfNotExists: 878337c82793). در این متد ردیابهای مورد نظر بروی خصوصیات-کنترلها و خصوصیت-روابط داده ایجاد می گردند. درباره این متد و کاربردهای آن در بخشهای بعدی توضیحات کاملتری ارائه می گردد.

پس از افزودن ردیابها با هر تغییر خصوصیت یا رابطه از شروط وضعیت به منبع پایش یا همان ابزار مدیریت پایش اطلاع داده می شود و به این عملیات سنجش وضعیتها و اعمال آنها مجدداً بروی داده انجام می گردد (UIActionActivityManager.DataPropertyRelationshipChanged: b60b5cf2effa).

##### کنترل وضعیتها داینامیک به هنگام تایید و ثبت داده :

مورد آخر استفاده از متد CheckAndImposeEntityStates یا همان سنجش و اعمال وضعیتها به هنگام تایید داده ها به منظور ثبت و در متد UpdateData می باشد(BaseEditEntityArea.UpdateData: 9376d8dbc2e0). به این ترتیب داده ها برای آخرین بار با شرط وضعیت کنترل می گردند. درباره این متد و جزئیاتش در بخشهای بعدی توضیحات ارائه خواهد شد.

#### متد AddChangeMonitorIfNotExists بروی داده :

در بخش قبل این متد معرفی شد و اینکه هدفش مدیریت ردیابهای لازم بروی داده می باشد(DP\_FormDataRepository.AddChangeMonitorIfNotExists: 878337c82793). امّا در این بخش به تشریح نحوه عملکرد و کاربردهایش می پردازیم. کاربرد اصلی این متد در مدیریت وضعیتها و همچنین مدیریت فرمولها به منظور پایش شروط مربوطه می باشد. به این متد به روی هر داده فرمی تعریف شده و ورودی آن یک ChangeMonitorItem می باشد که ساختار آن در زیر اشاره شده است.

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ChangeMonitorSource | I\_ChangeMonitor | منبع پایش |
| UsageKey | string | شناسه کلید |
| DataToCall | DP\_FormDataRepository | داده هدف |
| RelTailAndColumns | Tuple<string,List<int>> | رشته رابطه و شناسه ستونها |

خصوصیت ChangeMonitorSource همان منبع پایش می باشد که برای مثال ابزار مدیریت وضعیت و یا مدیریت فرمول مربوطه است. این خصوصیت از نوع I\_ChangeMonitor می باشد که این دارای متدی به نام DataPropertyRelationshipChanged به منظور اطلاع رسانی به منبع در مورد تغییر خصوصیات یا روابط داده هدف است. خصوصیت RelTailAndColumns نیز از دو بخش تشکیل شده است یکی رشته رابطه مورد پایش و دیگری در صورت وجود شناسه ستونهای مورد پایش. در متد AddChangeMonitorIfNotExists ابتدا کنترل می شود که آیتم مورد پایش برای داده تکراری نباشد. سپس این مورد به لیست موارد پایش افزوده می شود و اگر داده کامل باشد با ارسال مورد پایش متد SetChangeMonitor به منظور افزودن ردیابها به خصوصیت-کنترلها و روابط فراخوانی می شود (DP\_FormDataRepository.SetChangeMonitor: 66a9c49dd77a). همانطور که اشاره شد داده باید کامل باشد و در صورتی که داده نمایشی باشد طبیعتاً نیازی به افزودن ردگیرها بروی خصوصیات و روابط آن نمی باشد البته تا زمانی که داده به داده کامل تبدیل نشود، که در این صورت متد SetChangeMonitor فراخوانی خواهد شد.

#### متد SetChangeMonitor:

هدف این متد افزودن ردیابها به خصوصیت-کنترل های یک داده و یا داده های مرتبط می باشد. روال انجام شده در این متد به این شرح است.

* اگر رشته رابطه مورد پایش خالی باشد یعنی ستونهای همین داده موجود هدف پایش می باشند بنابراین به ازای تمامی ستونها خصوصیت-کنترل مربوطه شناسایی شده و رویداد PropertyValueChanged مورد استفاده قرار می گیرد. به این ترتیب با تغییر مقدار این خصوصیت از طریق مورد پایش مربوطه به منبع پایش تغییر مقدار این خصوصیت اطلاع داده می شود.
* اگر رشته روابط خالی نباشد به معنی اینست که هدف مورد پایش رشته رابطه ای از داده موجود می باشد. حال یا خصوصیت-رابطه های داده و یا رابطه داده با داده پدر خود. به هر حال مورد پایش جدیدی با حذف این رابطه جاری از رشته رابطه هدف ساخته می شود و :
  + اگر اولین رابطه مربوط به خصوصیت-رابطه های فرزند بود به ازای هر داده موجود در رابطه متد AddChangeMonitorIfNotExists با استفاده از مورد پایش جدید فراخوانی می شود. (اگر داده ای بعداً اضافه شود ردیابهای مرتبط به آن داده همان موقع اضافه خواهند شد)
  + اگر رابطه بین داده جاری و پدر بود متد AddChangeMonitorIfNotExists برای داده پدر با استفاده از مورد پایش جدید فراخوانی می شود.

نهایتاً با ایجاد ردیابهای لازم بروی داده هر گاه که خصوصیت یا روابط مورد پایش تغییر کنند رویداد DataPropertyRelationshipChanged برای منبع پایش فراخوانی می شود. بطور دقیق این رویداد در دو حالت فراخوانی می شود، با تغییر مقدار خصوصیت مورد پایش (DP\_FormDataRepository.PropertyValueChanged1: 955b8130c9b3) و یا با تغییر داده های خصوصیت-رابطه مورد پایش (ChildRelationshipInfo.CheckRelationshipChanged: e7c6d51dcb3d).

#### کاربردهای متد SetChangeMonitor:

همانطور که ذکر شد کاربرد این متد دو مورد می باشد:

* هنگام افزودن مورد پایش به یک داده کامل
* هنگام تبدیل یک داده نمایشی به داده کامل و پس از نمایش آن

#### موارد کاربرد متد AddChangeMonitorIfNotExists:

در این بخش موارد کاربرد متد مدیریت ردیابهای داده را بررسی می نماییم.

* در ابزار مدیریت وضعیتها و پس از نمایش داده در فرم که پیشتر اشاره شد (از طریق متد SetDataItemChangeMonitors)
* در ابزار مدیریت فرمولها و پس از نمایش داده
* در متد SetChangeMonitor به منظور افزودن ردیابها به داده های مرتبط که پیشتر اشاره شد
* در صورت افزودن داده به خصوصیت-رابطه از طریق متد AddDataToChildRelationshipData

کاربرد آخر برای داده هایی می باشد که داده پدر آنها برای خصوصیت-رابطه مرتبط دارای ردیاب می باشد اما داده در زمان افزودن ردیاب به داده پدر هنوز با داده پدر مرتبط نبوده است. به عبارت ساده تر برای داده هایی که بعداً توسط کاربر با داده پدر مرتبط می شوند (جدید افزوده می شوند و یا انتخاب می شوند) ردیاب از داده پدر افزوده می گردد.

#### وضعیتهای داینامیک خود ساخته در فرم ها :

از وضعیتهای داینامیک و آثارشان در عملیاتی کردن برخی نیازمندیهای سیستمی در فرمهای کار با داده ستفاده می شود. این وضعیتها در همان ابتدای ایجاد ابزار مدیریت وضعیتها برای هر فرم کار با داده توسط متد CreateCustomStates ساخته می شوند(UIActionActivityManager.CreateCustomStates: 9d99f631a921). در ادامه به تشریح این نیازمندیها و پیاده سازی آنها توسط وضعیتهای داینامیک پرداخته شده است.

##### فعال و غیر فعال شدن رابطه ارث بری بر اساس فیلد تعیین کننده:

در پیاده سازی رابطه های ارث بری گاهاً در موجودیت پدر فیلدهای تعیین کننده ای وجود دارد که بر اساس مقدار آن روابط بین پدر و فرزند فعال و یا غیر فعال می شوند. برای پیاده سازی این نیازمندی می توان در واسط کاربری از یک وضعیت با اثر عدم دسترسی به رابطه استفاده کرد. به این منظور از متد CheckISADeterminerStates استفاده می شود (UIActionActivityManager.CheckISADeterminerStates: 56344e195b36). در این متد رویداد زیر رخ می دهد :

* اگر در بین روابط فرزند فرم رابطه ارث بری پدر به فرزند وجود داشت (در واقع فرم جاری، موجودیت پدر، و روابط موجودیتهای فرزند باشند) و هر رابطه دارای فیلد تعیین کننده با مقادیر مشخص باشد، به ازای آن رابطه یک وضعیت در فرم جاری (فرم موجودیت پدر) تعریف می شود که شرط آن عدم وجود مقادیر تعیین کننده در ستون تعیین کننده باشد و اثر آن عدم دسترسی به رابطه مذکور. به این ترتیب اگر مقدار به جز مقادیر تعیین کننده انتخاب شود آن رابطه پدر به فرزند حذف می شود. طبیعتاً بر اساس شرط تعریف شده و ایجاد ردیاب بروی خصوصیت مقدار تعیین کننده با هر تغییر مقدار تعیین کننده وضعیت عدم دسترسی رابطه یکبار سنجش و اعمال می گردد.
* اگر فرم جاری از یک رابطه فرزند به پدر ارث بری بوجود آمده باشد و رابطه دارای فیلد تعیین کننده با مقادیر مشخص باشد، یک وضعیت در فرم جاری (فرم موجودیت پدر ارث بری) تعریف می شود که شرط آن عدم وجود مقادیر تعیین کننده در ستون تعیین کننده باشد و اثر آن عدم دسترسی به رابطه ریشه فرم. به این ترتیب اگر مقدار به جز مقادیر تعیین کننده انتخاب شود فرم جاری به علت عدم دسترسی به رابطه ریشه بی اثر می شود. در این حالت وضعیت دیگری نیز تعریف می شود که هدف آن مقداردهی خودکار فیلد تعیین کننده برای داده های جدید بصورت خودکار است. به این منظور یک وضعیت با شرط عدم برابری فیلد تعیین کننده با یک مقدار غیر ممکن تعریف می شود که همیشه شرط برقرار باشد. اثر وضعیت نیز مقداردهی فیلد تعیین کننده با اولین مقدار ممکن از لیست مقادیر است. به این ترتیب زمانی که موجودیت جاری از رابطه فرزند ارث بری به وجود آمده باشد به طور خودکار مقدار تعیین کننده متناسب رابطه انتخاب می شود.

##### فعال و غیر فعال شدن رابطه اتحاد بر اساس فیلد تعیین کننده:

در مورد روابط اتحاد و زیر اتحاد نیز دقیقاً به مانند بخش قبل عمل می شود (روابط ارث بری) و وضعیتی به منظور فعال و غیر فعال نمودن رابطه بر حسب فیلد تعیین کننده موجودیت اتحاد و همچنین تعیین مقدار این فیلد به صورت خودکار در متدی به نام CheckUnionDeterminerStates ایجاد می شود (UIActionActivityManager.CheckUnionDeterminerStates: 0199f3874e7c).

##### وضعیت فیلتر کردن بازه داده فیلدها:

برای پیاده سازی نیازمندی فیلتر شدن بازه داده در صورت وجود در فرم، یک وضعیت با شرطی بر اساس ستون و رشته رابطه فیلتر بازه داده ایجاد می گردد. البته این شرط همواره برقرار است و امّا باعث می شود با ایجاد ردیاب تغییرات بروی ستون مقدار فیلتر با هر تغییری در این مقدار وضعیت مجدداً سنجش و اعمال شود.اثر وضعیت نیز طبیعتاً از نوع فیلتر بازه ایجاد می گردد. ستون هدف همان ستون دارای بازه و مقدار فیلتر این اثر همان ستون و رشته رابطه فیلتر ستون دارای بازه می باشد. این رویداد در متدی به نام CheckColumnValueRangeStates انجام می شود (UIActionActivityManager.CheckColumnValueRangeStates: 403d341eba76).

## عملیات زمینه ای بروی داده :

یکی از قابلیتهای سامانه ورود اطلاعات امکان تعریف عملیاتهای زمینه ای و پشت صحنه بروی داده ها قبل یا بعد از تراکنشهای مشخص داده به مانند واکشی داده، ثبت داده و یا حذف داده ها می باشد. به این ترتیب قبل یا بعد از هر تراکنش امکان فراخوانی رویدادهای خارج از سیستم به مانند قطعه کد و یا یک رویداد دیتابیسی بصورت مستقیم فراهم می گردد. ساختار قابل تعریف هر عملیات به شرح زیر می باشد:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| Title | string | عنوان عملیات |
| EntityID | int | شناسه موجودیت مرتبط |
| Type | Enum\_ActionActivityType | نوع عملیات |
| CodeFunctionID | int | شناسه قطعه کد |
| CodeFunction | CodeFunctionDTO | موجودیت قطعه کد |
| DatabaseFunctionEntityID | int | شناسه رویداد دیتابیس و ارتباط با موجودیت |
| DatabaseFunctionEntity | DatabaseFunction\_EntityDTO | رویداد دیتابیس و ارتباط با موجودیت |
| Step | Enum\_EntityActionActivityStep | مرحله اجرای عملیات |
| ResultSensetive | ResultSensetive | حساسیت به نتیجه |

**خصوصیت شناسه موجودیت:**

نکته مهم درباره تعریف عملیاتهای زمینه ای اینست که مرتبط کردن آنها با یک موجودیت الزامی نیست. در واقع با مشخص نکردن موجودیت مرتبط امکان عمومی سازی اجرای آن عملیات بروی تمامی موجودیتها وجود دارد.

**خصوصیت مرحله اجرای عملیات (Step):**

این خصوصیت مقطعی را نشان می دهد که می بایست عمیلیات بروی داده انجام شود. مقادیر قابل تعریف برای این خصوصیت پنج مرحله زیر می باشد:

* بعد از واکشی داده (AfterDataFetch)
* قبل از ذخیره داده (BeforeSave)
* بعد از ذخیره داده (AfterSave)
* قبل از حذف داده (BeforeDelete)
* بعد از حذف داده (AfterDelete)

درباره نحوه اجرای هر کدام از این مراحل در بخشهای بعدی توضیحاتی ارائه خواهد شد. بخش مهم عملیات تعیین جزئیات اجرای عملیات بروی داده می باشد. نوع آن که در خصوصیت Type تعیین می شود یا قطعه کد است (CodeFunction) و یا یک تابع دیتابیس (DatabaseFunction) که بسته به مرحله عملیات بروی داده قابل اجرا می باشند. در ادامه به نحوه تعریف یک عملیات و معرفی رویداد مرتبط پرداخته می شود.

### نحوه تعریف عملیات زمینه ای:

عملیاتهای زمینه ای در برنامه مدیریت فراداده و در فرمی مخصوص تعریف می شوند(frmBackendActionActivity: dc969e183ee4). حال ممکن است این عملیات مختص یک موجودیت خاص باشد و یا بدون شناسه موجودیت بصورت عمومی تعریف شود. در این فرم ضمن تعیین عنوان و مرحله اجرای عملیات، رویداد اجرایی از جنس قطعه کد یا تابع دیتابیس یا از لیست موجود انتخاب شده و یا بصورت میانبر در فرم های مخصوص ایجاد می گردند. در ادامه به نحوه تعریف رویدادها پرداخته می شود.

### تعریف قطعه کد (CodeFunction):

قطعه کد ها رویدادهای موجود در کتابخانه های مستقلی می باشند که بصورت ضمنی در مدل فراداده تعریف می شوند و در کاربردهای مختلف از آنها استفاده می شود. این قطعه کدها در فرمی مخصوص تعریف می شوند(frmCodeFunction: 6edc06fffe32). ساختار هر موجودیت قطعه کد به شرح زیر است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| Name | string | نام قطعه کد |
| Path | string | مسیر فایل |
| ClassName | string | نام کلاس |
| FunctionName | string | نام رویداد |
| RetrunType | string | رشته نوع مقدار برگشتی |
| RetrunDotNetType | Type | نوع مقدار برگشتی در دات نت |
| ParamType | Enum\_CodeFunctionParamType | نوع کلی پارامتر ورودی رویداد |
| Parameters | List<CodeFunctionColumnDTO> | پارامترهای ورودی رویداد |
|  |  |  |

در فرم تعریف قطعه کدها، فایل کتابخانه حاوی قطعه کد مشخص می شود و سپس از لیست کلاسها و رویدادهای فایل سایر جزئیات قطعه کد به مانند نوع مقدار برگشتی و لیست پارامتر ها مشخص می شوند. خصوصیت ParamType نوع قطعه کد را از منظر کلی پارامترهای ورودی مشخص می کند. پارامتر های ورودی قطعه کد ها می توانند به طور کلی از چند نوع باشند:

* ManyDataItems : چند داده ای، پارامتر ورودی قطعه کد می بایست از نوع CodeFunctionParamManyDataItems باشد که شامل لیستی از داده کامل یا همان DP\_DataRepository باشد.
* OneDataItem : تک داده ای، تک پارامتر ورودی قطعه کد می بایست از نوع CodeFunctionParamOneDataItem باشد که شامل تک داده کامل یا همان DP\_DataRepository باشد.
* SelectedColumns : ستونها انتخابی، پارامترهای ورودی قطعه کد می توانند بیش از یک مورد باشند امّا می بایست همگی از انواع پایه (Primitive) باشند. تناظر بین این پارامترها و خصوصیتات یک داده در فرمی دیگر تعریف می شود که کاربرد آن بعداً اشاره خواهد شد.
* LetterFunction : موجودیت نامه ، تک پارامتر ورودی قطعه کد می بایست از نوع موجودیت نامه باشد.

لازم به یادآوری است که قطعه کدها خود کاربردی مستقل ندارند و بر اساس نیازمندیهای از آنها در بخش های مختلف برنامه استفاده می شود. بنابراین نوع کلی پارامترهای ورود قطعه کد از بیرون فرم تعریف آنها و بر اساس کاربرد مشخص می شود. برای مثال در صورتی که در عملیات زمینه ای در مرحله بعد از واکشی داده قطعه کد مورد نیاز باشد باید حتماً پارامتر ورودی آن از نوع چند داده ای باشد. در فرم تعریف قطعه کد نیز قبل از ثبت آن اعتبارسنجی می شود که آیا پارامترهای ورودی قطعه کد با نوع کلی آن هماهنگ است یا خیر (frmCodeFunction.CheckMethodValidity: b4b3c26ca3d9). سپس قطعه کد برای استفاده های بعدی ذخیره می شود (BizCodeFunction.22a0734b5c8e: 22a0734b5c8e). همچنین برای دریافت لیست قطعه کد ها با توجه به کاربرد و نوع کلی پارامترهای مورد انتظار از متد GetAllCodeFunctions استفاده می شود (BizCodeFunction.GetAllCodeFunctions: 26a81b546118). کاربردهای قطعه کدها در مواردی به مانند عملیاتهای زمینه ای، فرمولها و دکمه های اختصاصی می باشد. در ادامه به نحوه اجرای قطعه کدها بروی داده ها می پردازیم.

#### اجرای قطعه کد:

بطور کلی اعمال قطعه کد ها در ابزاری مخصوص به نام CodeFunctionHandler انجام می شود(CodeFunctionHandler: d1ee92e5176f). اعمال قطعه کد بنابر نوع کلی پارامتر آن متفاوت می باشد:

##### اعمال قطعه کد چند داده:

برای اجرای قطعه کد چند داده (که به طور نمونه در عملیات زمینه ای واکشی داده نیز به آن اشاره شد) از متد مخصوص استفاده می شود(CodeFunctionHandler.GetCodeFunctionResultMultipleDataItems: ad4b85c0fbe9). پارامتر دقیق پیاده سازی شده قطعه کد می بایست از نوع CodeFunctionParamManyDataItems باشد که از اطلاعات کاربر جاری و لیست داده ها تشکیل می شود. نهایتاً این پارامتر به متد اصلی اجرای قطعه کد یا همان GetCodeFunctionResult فرستاده می شوند

##### اعمال قطعه کد تک داده:

برای اجرای قطعه کد تک داده (که به طور نمونه در عملیات زمینه ای ثبت داده کاربرد دارد) از متد مخصوص استفاده می شود(CodeFunctionHandler.GetCodeFunctionResultOneDataItem: 2f03f2a5f5bf). پارامتر دقیق پیاده سازی شده قطعه کد می بایست از نوع CodeFunctionParamOneDataItem باشد که از اطلاعات کاربر جاری و تک داده تشکیل می شود. نهایتاً این پارامتر به متد اصلی اجرای قطعه کد یا همان GetCodeFunctionResult فرستاده می شوند

##### اعمال قطعه ستونهای انتخابی:

...

##### متد اصلی اجرای قطعه:

برای اجرای نهایی قطعه کد از متد مخصوصی استفاده می شود(CodeFunctionHandler.GetCodeFunctionResult: be5fc56a3296). در این متد با استفاده از مسیر کتابخانه، نام کلاس و نام تابع هدف، قطعه کد با ارسال پارامترهای مورد انتظار فراخوانی می گردد(ReflectionHelper.CallMethod :30fe1df83ac0).

در ادامه به بررسی نحوه تعریف و کاربرد قطعه کد ها تنها در عملیاتهای زمینه ای می پردازیم و در بخشهای دیگر به سایر کاربردهای قطعه کدها اشاره خواهد شد.

### ارتباط قطعه کدها و عملیات زمینه ای:

از قطعه کدهای تعریف شده میتوان در عملیاتهای زمینه ای یک موجودیت استفاده نمود. بسته به این که عملیات مربوط به چه مرحله ای از ارتباط با داده باشد ممکن است نوع قطعه کد قابل تعریف و انتخاب از منظر نوع کلی پارامتر ورودی متفاوت باشند. در واقع برای عملیاتهای بعد از واکشی داده چون با لیستی از داده های واکشی شده مواجه هستیم، قطعه کد می بایست از نوع پارامتر چند داده (ManyDataItems) باشد. برای سایر مراحل که بعد از قبل عملیاتهای حذف را شامل می شود قطعه کدهای تک داده (OneDataItem) مورد استفاده می باشد. تعیین نوع قطعه کد مورد استفاده با توجه به مرحله عملیات زمینه ای، در متد GetCodeFunctionParamType انجام می شود (frmBackendActionActivity.GetCodeFunctionParamType: e6ae2a796812). لیست قطعه کدها نیز با توجه به این پارامتر نوع از همان متد GetAllCodeFunctions دریافت می گردد.

### تعریف تابع پایگاه داده-موجودیت (DatabaseFunction) :

در مباحث ابتدایی عنوان شد که توابع پایگاه داده را نیز می توان از پایگاه داده هدف استخراج نموده و در مدل فراداده نگهداری نمود. سپس از این توابع را می توان به بصورت مستقل و در کاربردهای مختلف فراخوانی نمود. از آنجایی که پارامترهای ورودی توابع پایگاه داده ممکن است متنوع و متعدد باشند امکان استفاده مستقیم از این توابع وجود ندارد و نیاز به یک موجودیت واسط به منظور تعیین ارتباطات پارامترهای تابع و ستونهای موجودیت ها می باشد. بنابراین برای هر موجودیت بسته هایی قابل تعریف می باشند که شامل عنوان ارتباط موجودیت-تابع و ترکیب ستونها و پارامترهای تابع به منظور ارسال به تابع هنگام فراخوانی تابع می باشند. سپس از این بسته ها در کاربردهای مختلف به مانند عملیاتهای زمینه ای استفاده می شود. این موجودیتهای واسط در فرم مخصوصی در برنامه مدیریت فراداد تعریف می شوند (frmDatabaseFunction\_Entity: 5ae0fa3f28be). هر واسط تابع-موجودیت از نوع DatabaseFunction\_EntityDTO می باشند که به ساختار آنها در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| Name | string | نام قطعه کد |
| Title | string | مسیر فایل |
| EntityID | int | نام کلاس |
| DatabaseFunctionID | int | شناسه تابع پایگاه داده |
| DatabaseFunction | DatabaseFunctionDTO | موجودیت تابع پایگاه داده |
| DatabaseFunctionEntityColumns | List<DatabaseFunction\_Entity\_ColumnDTO> | لیست ارتباط ستونها و پارامترها |

هر DatabaseFunction\_Entity\_ColumnDTO نیز دارای ساختار زیر می باشد:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| DBFunctionParameterID | string | شناسه پارامتر تابع |
| ColumnID | string | شناسه ستون |
| DBFunctionParamDotNetType | int | نوع دات نت پارامتر |

بطور کلی لیست این واسطها برای یک موجودیت از متد GetDatabaseFunctionEntities دریافت می شود(BizDatabaseFunction.GetDatabaseFunctionEntities: 5c860f83a5fd). در تعریف عملیاتهای زمینه ای بسته به نوع مرحله مربوطه عملیات می توان از واسط های تعریف شده بروی موجودیت استفاده نمود. برای مراحل بعد از واکشی داده و قبل از ثبت و حذف داده استفاده از توابع پایگاه داده امکان پذیر نمی باشد و تنها قطعه کد مورد استفاده است. برای سایر عملیاتها یعنی بعد از ثبت و حذف داده استفاده از واسط تابع پایگاه داده امکان پذیر است. به منظور فراخوانی تابه پایگاه داده با استفاده از این واسط ها از متد GetDatabaseFunctionValue ) استفاده می شود (DatabaseFunctionHandler.GetDatabaseFunctionValue: b473b946c026)

کاربردهای دیگر توابع پایگاه داده در تعریف فرمولها می باشد که در بخشهای دیگر به آن اشاره خواهد شد.

### نحوه اعمال عملیاتهای زمینه ای:

نحوه اعمال عملیاتهای زمینه ای به مرحله آن بستگی دارد. در هر مرحله، لیست عملیاتهای قابل اجرا با ارسال نوع مرحله و شناسه موجودیت از متد GetBackendActionActivities دریافت می شود.(BizBackendActionActivity.GetBackendActionActivities: 6a4380912411). لازم به ذکر است که علاوه بر عملیاتهای مخصوص موجودیت، عملیاتهای عمومی و بدون موجودیت نیز به منظور اجرا به لیست افزوده می شوند. سپس هر عملیات بر حسب اینکه قطعه کد است یا تابع پایگاه داده در همان مرحله بروی داده ها اعمال می شوند. البته همانطور که ذکر شد برخی مراحل عملیات تنها مختص اجرای قطعه کدها می باشد و نه توابع پایگاه داده. در ادامه نحوه اجرای عملیاتهای هر مرحله اشاره شده است:

#### اعمال عملیات بعد از واکشی داده :

عملیاتهای تعریف شده زمینه ای بعد از واکشی (AfterDataFetch)، در همان متد واکشی داده های کامل( ProcessSearchEditRequest) از طریق متد مخصوص DoAfterDataFetchBackendActionActivities انجام می شود (SearchRequestProcessor.DoAfterDataFetchBackendActionActivities: c263cf868047). در این متد پس از دریافت لیست مخصوص عملیاتهای این مرحله از متد GetBackendActionActivities، هر عملیات بروی داده های واکشی شده اعمال می شود که همانطور که ذکر شد تنها از نوع قطعه کد می باشند. بنابراین متد GetCodeFunctionResultMultipleDataItems که پیشتر تشریح گردید بروی داده ها اعمال می شود.

#### اعمال عملیات قبل از ثبت داده ها و حذف داده ها :

عملیاتهای نهایی ثبت داده و حذف داده در دو حالت ساخته می شوند، یکی درخواست ثبت داده ها (ProcessEditRequest) و دیگری درخواست حذف داده (ProcessDeleteRequest). در هر دو حالت لیست دستور-داده (QueryItem) ها تولید می شوند که ممکن است شامل موارد ثبت جدید، بروزرسانی و حذف باشند. برای در خواست ثبت داده ممکن است درخواست های حذف داده نیر همراهش ارسال شوند و برای درخواست حذف داده ممکن است نیاز به بروزرسانی کلید خارجی داده وابسته به مقدار Null باشد. برای انجام عملیاتهای زمینه ای پیش از ثبت و حذف داده از یک متد مشترک به نام DoBeforeEditActionActivities استفاده می شود که ورودی آن لیست اقلام دستور-داده از هر نوع ثبت یا حذف می باشد (EditDataActionActivityManager.DoBeforeEditActionActivities: d451b5c0ef6e). سپس در متد بسته به اینکه دستور-داده متعلق به چه موجودیت و از چه نوعی است، عملیاتهای زمینه ای مناسب بروی آن انجام می شود. روال انجام شده در این متد به شرح زیر است:

* به ازای هر قلم داده از لیست تغییرات پایگاه داده هدف (که می تواند عملیات ثبت جدید، بروزرسانی و یا حذف باشد):
  + اگر ثبت جدید و یا بروزرسانی بود، لیست عملیاتهای زمینه ای پیش از ثبت دریافت می شود
  + اگر حذف بود لیست عملیاتهای زمینه ای پیش از حذف دریافت می شود
  + به ازای هر مورد در لیست عملیاتهای زمینه :
    - ای آن عملیات اجرا می شود (که تنها از نوع قطعه کد می باشند)
    - شناسه عملیات و خطای احتمالی (در صورت وجود) به لیست مخصوص عملیاتهای پیشین برای قلم داده اضافه می شود (از این اطلاعات بعداً و در خارج از متد در ثبت لاگ دستور-داده استفاده می شود).
    - اگر خطایی رخ داده باشد نتیجه کلی عملیات قبل از ثبت منفی برگردانده می شود
    - لاگ مربوطه ذخیره می شود
  + اگر خطایی در عملیاتهای قلم داده رخ داده باشد لاگ مربوطه به نتیجه نهایی عملیات درخواست ثبت یا حذف به منظور نمایش در واسط کاربری افزوده می شود

#### اعمال عملیات پس از ثبت و حذف داده ها :

پس از انجام موفق کلیه تراکنشها در درخواستهای ثبت و حذف داده ها بنابر همان دلایل ذکر شده در بخش قبل از یک متد مشترک به نام DoAfterEditActionActivities استفاده می شود. ورودی این متد نیز لیست اقلام دستور-داده از هر نوع ثبت یا حذف می باشد (EditDataActionActivityManager.DoAfterEditActionActivities: 349e5b1092af). سپس در متد بسته به اینکه دستور-داده متعلق به چه موجودیت و از چه نوعی است، عملیاتهای زمینه ای مناسب بروی آن انجام می شود. روال انجام شده در این متد به شرح زیر است:

* به ازای هر قلم داده از لیست تغییرات پایگاه داده هدف (که می تواند عملیات ثبت جدید، بروزرسانی و یا حذف باشد):
  + اگر ثبت جدید و یا بروزرسانی بود، لیست عملیاتهای زمینه ای پیش از ثبت دریافت می شود
  + اگر حذف بود لیست عملیاتهای زمینه ای پیش از حذف دریافت می شود
  + به ازای هر مورد در لیست عملیاتهای زمینه :
    - ای آن عملیات اجرا می شود (که ممکن است از نوع قطعه کد باشد و یا تابع پایگاه داده)
    - شناسه عملیات و خطای احتمالی (در صورت وجود) به لیست مخصوص عملیاتهای پسین برای قلم داده اضافه می شود (از این اطلاعات بعداً و در خارج از متد در ثبت لاگ دستور-داده استفاده می شود).
    - اگر خطایی رخ داده باشد نتیجه کلی عملیات پس از ثبت منفی برگردانده می شود
  + اگر خطایی در عملیاتهای قلم داده رخ داده باشد لاگ مربوطه به نتیجه نهایی عملیات درخواست ثبت یا حذف به منظور نمایش در واسط کاربری افزوده می شود

## سایر کاربردهای قطعه کد ها:

همانطور که تشریح شد یکی از کاربردهای قطعه کدها در تعریف عملیات های زمینه ای می باشد که برای عملیات بعد از واکشی داده بصورت چند داده و برای سایر عملیاتها بصورت تک داده استفاده می شوند. در ادامه به بررسی کاربردهای دیگر قطعه کدها و انواع دیگر آنها پرداخته می شود.

### کاربردهای دیگر قطعه کد تک داده :

* قطعه کد تک داده در تعریف دکمه های اختصاصی فرمها نیز استفاده می شود(frmEntityCommand: f7d70c177b8c). درباره این مفهوم در بخش مخصوص آن شرح داده خواهد شد.
* کاربرد دیگر قطعه کد تک داده در تعریف فرمول است زمانی که فرمول مستقیماً از نوع کد تابع می باشد(frmFormula: 893137ad1c61). در این حالت برای محسبه فرمول (FormulaFunctionHandler.CalculateFormula: c7e4de07ae7a) داده ای که فرمول روی آن اجرا می شود مستقیماً به قطعه کد ارسال شده و نتیجه به عنوان مقدار فرمول برگردانده می شود. در این مورد در بخش فرمولها توضیحاتی ارائه خواهد شد.

### قطعه کد با پارامتر ستونهای انتخابی:

یکی از انواع قطعه کدها، پیاده سازی با پارامتر ستونهای انتخابی می باشد که با نوع Enum\_CodeFunctionParamType.SelectedColumns متمایز می شود . در این حالت نیاز است که ابتدا این ارتباط ستونهای یک موجودیت و قطعه کد مرتبط در قالب یک موجودیت واسط تعریف شده و بجای ارسال کد آبجکت داده به قطعه کد، لیستی از این موجودیت واسط برای استخراج ستونهای داده و ارسال آنها به عنوان پارامتر قطعه کد استفاده شود. این موجودیت واسط به عنوان CodeFunction\_EntityDTO می باشد و دارای ساختار زیر است :

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| EntityID | int | شناسه موجودیت مرتبط |
| CodeFunctionID | int | شناسه قطعه کد مرتبط |
| Title | string | عنوان |
| Name | string | نام |
| CodeFunctionEntityColumns | List<CodeFunction\_Entity\_ColumnDTO> | ستونهای درگیر |

لیست ستونهایی که به عنوان پارامتر به قطعه کد ارسال خواهند شد در خصوصیت CodeFunctionEntityColumns می باشند. هر مورد از نوع CodeFunction\_Entity\_ColumnDTO است و دارای ساختار زیر است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| CodeFunctionParameterID | int | شناسه پارامتر قطعه کد |
| ColumnID | int | شناسه ستون |

کاربرد این موجودیت واسط زمانی است که فرمول مستقیماً از آن نوع می باشد. در این حالت به هنگام محاسبه فرمول (FormulaFunctionHandler.CalculateFormula: c7e4de07ae7a داده ای که فرمول روی آن اجرا می شود به همراه واسط قطعه کد به متد GetCodeFunctionEntityResult به منظور اجرا و تعیین نتیجه فرمول ارسال می شود (CodeFunctionHandler.GetCodeFunctionEntityResult: f1bb53724318) در این مورد در بخش فرمولها توضیحات بیشتری ارائه خواهد شد.

### قطعه کد با پارامتر نامه:

یکی دیگر از انواع قطعه کد به منظور اعمال بروی موجودیت نامه ها تعریف شده است. کاربرد اصلی آن در تعریف تنظیمات نامه و به منظور تعریف عملیاتهای زمینه ای قبل از ذخیره شده نامه، بعد از ذخیره شدن نامه، تبدیل نامه و منبع خارجی نامه می باشد. در این مورد در بخش مربوطه توضیحات بیشتری ارائه شده است. برای اعمال این قطعه کد ها در مقطع مورد نیاز از متد GetCodeFunctionLetterResult استفاده می شود (CodeFunctionHandler.GetCodeFunctionLetterResult: a3b47f316b22).

## سایر کاربردهای تابع پایگاه داده :

کاربرد دیگر توابع پایگاه داده استفاده از واسطه های توابع تعریف شده (که پیشتر اشاره شد) در تعریف فرمول است زمانی که فرمول مستقیماً از نوع واسط تابع پایگاه داده باشد می باشد. در این حالت نیز برای محاسبه فرمول (CalculateFormula) از واسط تابع و داده فرمول استفاده می شود و مقدار نتیجه تابع پایگاه داده توسط متد GetDatabaseFunctionValue دریافت و به عنوان نتیجه فرمول ارسال می شود.

همچنین در کاربردی دیگر تمامی واسطهای توابع یک موجودیت، به عنوان خصوصیات فرمول های خطی می باشد (FormulaInstanceInternalHelper.GetProperties: 40d8aa25e83f).

درباره کاربرد واسط های توابع پایگاه داده در فرمولها در بخش مربوطه توضیحاتی ارائه خواهد شد.

## فیلتر داده های رابطه در فرم ورود اطلاعات :

یکی از قابلیتهای فرمهای ورود اطلاعات امکان تعریف و اعمال فیلتر برای جستجوی داده های یک رابطه می باشد. به عنوان مثال فرض کنید در حال ورود اطلاعات یک داده هستیم. برای یک رابطه از داده نیاز داریم تا داده طرف دیگر را جستجو کنیم و انتخاب نماییم، امّا نمی خواهیم تمام داده های موجودیت طرف دیگر قابل جستجو و انتخاب باشند، بلکه تنها داده هایی قابل جستجو باشند که ارتباطی با داده جاری و داده های مرتبط با داده جاری دارند. برای فیلترهای داده های رابطه از موجودیتی به نام RelationshipFilterDTO استفاده می شود:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | int | شناسه |
| RelationshipID | int | شناسه رابطه |
| Title | string | عنوان فیلتر |
| ValueRelationshipTailID | int | شناسه رشته رابطه مقدار |
| ValueColumnID | int | شناسه ستون مقدار |
| SearchRelationshipTailID | int | شناسه رشته رابطه شرط جستجو |
| SearchColumnID | int | شناسه ستون شرط جستجو |

برای تعریف فیلترها برای یک رابطه از فرمی مخصوص استفاده می شود(frmRelationshipFilter: 272cb9095085). در این فرم فیلترها پس از تعریف و انتخاب ستونها و رشته روابط هدف فیلتر، در مدل فراداده ذهیره می شوند (BizRelationshipFilter.UpdateRelationshipFilters: f695b99dcea8).

در فرمهای ورود اطلاعات در صورتی که فرم از رابطه بر آمده باشد از خصوصیت RelationshipFilters به منظور دسترسی به فیلترهای رابطه استفاده می شود(BaseEditEntityArea.RelationshipFilters: ab2ed0528c76). به منظور اعمال آنها نیز از این فیلترها در فرم های ورود اطلاعات و در همان متد GetDataViewItemsFromSearchResult استفاده می شود (BaseEditEntityArea.GetDataViewItemsFromSearchResult: 7e4651a61540) که وظیفه اش انجام جستجوی اعمال شده در فرم ها می باشد. به این ترتیب به عبارت جستجو، به ازای هر فیلتر یک عبارات جستجو برگرفته شده از رشته رابطه و ستون شرط جستجو (SearchColumnID) با مقدار رشته رابطه و ستون فیلتر افزورده می شود(ValueColumnID).

کاربرد دیگر این فیلترها دراعتبارسنجی داده های انتخاب شده در یک رابطه می باشد تا با فیلترهای تعریف شده در توافق باشند(UIValidationManager.ValidateRelationshipFilters: 8b0044b51038). برای این سنجش یک عبارت جستجو شامل کلیدهای تمامی داده های مرتبط با داده اصلی و اعمال فیلتر ساخته می شود و توسط متد SendSearchKeysOnlyRequest جستجو می شود. حال هر داده ای در رابطه که در نتایج این جستجو موجود نبود نشان از عدم برقراری فیلتر بروی آن دارد و با نمایش پیغامی مناسب در واسط کاربری به عنوان عدم اعتبارسنجی فیلتر رابطه مشخص می شود.

## نمای ظاهری داده ها:

یکی از قابلیتهای سامانه، نمایش داده های جستجو شده در نماهای مختلف ظاهری می باشد. این نما می تواند یک جدول داده باشد و یا باکس های گرافیکی. در این نماها می توان برای هر داده به قابلتهای دیگر سامانه (مانند آرشیو، نامه ها، فرمها و گزارشات) بصورت میانبر دسترسی داشت. همچنین می توان بین داده های مرتبط یک داده در قالب نمای دیگری بصورت تودرتو پیمایش نمود. به این جزئیات در بخشهای بعدی پرداخته خواهد شد. امّا برای فراخوانی فرم نمای داده از متد ShowDataViewGridViewArea استفاده می شود (AgentUICoreMediator.ShowDataViewGridViewArea: 564d6ae87497). به این متد پارامترهایی به مانند شناسه موجودیت مرتبط، عبارت پیش فرض جستجوی داده ها و داده قبلی و نمای مربوطه آن (در صورت وجود) ارسال می شود. در صورتی که نمای درخواستی حاصل از نمای دیگری نباشد (نمای ریشه باشد)، لازم است تا یک DataViewAreaContainer ایجاد شود که مسئولیت کنترل مسیرهای نمای اصلی را دارد(DataViewAreaContainer: d5ab5f7eafbd). برای ایجاد این کانتینر از DataViewAreaContainerInitializer استفاده می شود که ساختار آن در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| EntityID | int | شناسه موجودیت مرتبط |
| Title | string | عنوان نما |
| DataViewOrGridView | bool | نمای گرید یا باکس |
| InitialSearchRepository | DP\_SearchRepositoryMain | عبارت جستجوی پیش فرض |
| UserCanChangeSearchRepository | bool | آیا کاربر می تواند عبارت جستجو جدید ایجاد کند. |
| ShowInitializeSearchRepository | bool | عبارت جستجو پیش فرض همان ابتدا اعمال شود؟ |
| DataMenuSettingID | int | شناسه تنظیمات منو |

### مفهوم DataViewAreaContainer:

هدف از ایجاد DataViewAreaContainer واسط کاربری است که فرم نمای داده های جستجو شده و در صورت پیمایش، فرمهای نمای داده های مرتبط آنها در یک فضا بصورت متمرکز نمایش داده شوند. از سه بخش مهم تشکیل شده است، یکی بخش جستجوی داده ها، دیگری فضای نمایش فرمهای نمای داده ها (از جنس فرم نمای جدولی و یا فرم نمای باکسی) و نهایتاً پنل نمایش رشته سوابق پیمایشهای انجام شده. برای جستجوی داده ها از یک فرم جستجوی عمومی یا همان GeneralEntitySearchArea استفاده می شود(GeneralEntitySearchArea: 33b533adf734). بخش اصلی فرم جستجوی عمومی دقیقاً همان I\_SearchEntityArea است که در فرم های ورود اطلاعات استفاده می شود. تنها مورد اضافه در فرم عمومی اینست که شناسه موجودیت مورد جستجو اختیاری است. اگر شناسه موجودیت موجود نبود قابلیت جستجو و انتخاب موجودیتها مورد جستجو توسط کاربر وجود دارد. اگر شناسه موجودیت پیش فرض نیز وجود داشت که بصورت خودکار انتخاب می شود. در کاربردهای فرم جستجو عمومی به مانند فرم کانتینر نمای داده نیز ممکن است برخی مواقع شناسه موجودیت موجود نباشد و لازم باشد موجودیت بصورت عمومی توسط کاربر انتخاب شود. به هر حال در کانتینر نمای داده صورت انجام جستجوی داده توسط فرم جستجوی عمومی متد GeneralEntitySearchArea\_SearchDataDefined فراخوانی می شود که در آن اولین فرم نمای داده ایجاد و نمایش داده شده و عبارت جستجو نیز به آن فرم نما فرستاده می شود که قابلیتهای اصلی نمایش داده ها در آن انجام می گردد. برای افزودن هر فرم نمای داده از متد AddDataViewArea استفاده می شود (DataViewAreaContainer.AddDataViewArea: 3e2a0df94502). همانطور که ذکر شد نیز برای یک داده در یک نمای داده می توان به داده های مرتبطش در فرم نمای دیگر پرش نمود. تمامی این نماهای مرتبط در همان کانتینر اصلی نمایش داده می شوند. لینک سابقه این پیمایشها در کانتتینر و توسط متد SetLinks مدیریت می شود (DataViewAreaContainer.SetLinks: 214a299796b1). در ادامه به مفهوم فرمهای نمای داده و جزئیات آن پرداخته می شود.

### فرم نمای داده :

فرمهای نمای داده دارای یک کلاس مادر می باشند به نام DataArea که در آن فعالیتهای مشترک انواع فرمهای نمای داده به مانند جستجوی داده، تعیین لیست های نمایشی نمای داده ها، تعیین ترتیب داده ها و ... انجام می شود(DataArea : fc9a3ffeda2a). برای ایجاد نماهای داده از متد SetAreaInitializer استفاده می شود که به آن یک DataViewAreaInitializer ارسال می شود (DataArea.SetAreaInitializer: ab8f1a14b83a). ساختار هر DataViewAreaInitializer در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| EntityID | int | شناسه موجودیت مرتبط |
| Title | string | عنوان نما |
| DataMenuSettingID | int | شناسه تنظیمات منو |

در این کلاس متد GetDataItemsBySearchRepository نیز وظیفه اعمال عبارت جستجو و دریافت داده ها را دارد. امّا رویدادهای تخصصی فرمهای نمای داده به مانند افزودن داده به کنترلهای نمایشی در فرمهای تخصصی انجام می شود که در ادامه تشریح داده شده اند:

#### فرم نمای جدولی داده ها (GridViewArea):

برای ایجاد یک فرم نمای جدولی داده ها از متد SetAreaInitializerSpecialized استفاده می شود (GridViewArea.SetAreaInitializerSpecialized: dbedc44acc58). واسط کاربری این فرم به مانند همان فرم نمایش داده در فرم های ورود اطلاعات می باشد. تنها یک رویداد نسبت به آن افزون دارد به نام InfoClicked که در واقع در صورت انتخاب داده در جدول و کلیک بروی دکمه اطلاعات فرم به منظور نمایش منوی داده فراخوانی می شود. داده ها توسط متد SetItems به نمای داده اضافه می شوند که در واقع هر کدام تبدیل به یک ردیف جدول می گردند(GridViewArea.SetItems: 26be2f98a31d)

#### فرم نمای باکسی داده ها (DataViewArea):

برای ایجاد یک فرم نمای باکسی داده ها از متد SetAreaInitializerSpecialized استفاده می شود (DataViewArea.SetAreaInitializerSpecialized: 1bf8ccdeb8b2). داده ها توسط متد SetItems به نمای داده اضافه می شوند که در واقع هر کدام تبدیل به یک باکس گرافیکی می شوند (DataViewArea.SetItems: 2d5a40e5e8a0). روی هر باکس دکمه اطلاعات قرار دارد که در صورت کلیک منوی داده ایجاد و نمایش داده می شود.

### کاربردهای نمای ظاهری داده ها:

فرمهای نمای داده درون کانتینر نمای داده معنی پیدا می کنند که همانطور که اشاره شد این کانتینر در متد ShowDataViewGridViewArea ایجاد می شود. بنابراین برای بررسی کاربردهای نمای داده می توان کاربردهای این متد را بررسی نمود.

* با کلیک بروی دکمه نمای باکسی داده ها در فرمهای ورود اطلاعات (DataViewCommand: 939641817cea)
* با کلیک بروی دکمه نمای جدولی داده ها در فرمهای ورود اطلاعات (GridViewCommand: a7111982b91c)
* با کلیک بروی منوی نمای باکسی داده ها در فرم اصلی برنامه (AgentUICoreMediator.GridViewMenu\_Clicked: 2fb728ad84e4)
* با کلیک بروی منوی نمای جدولی داده ها در فرم اصلی برنامه (AgentUICoreMediator.GridViewMenu\_Clicked: b3e382bbfc54)
* با کلیک بروی منوی نمای باکسی یک موجودیت در درخت منوی برنامه (AgentUICoreMediator.NavigationMenuClicked: a3a9f1a1fdd4)
* با کلیک بروی منوی نمای جدولی یک موجودیت در درخت منوی برنامه (AgentUICoreMediator.NavigationMenuClicked: a3a9f1a1fdd4)
* در صورت فراخوانی متد نمایش گزارش جستجو محوری که از جنس نمای باکسی داده ها باشد (AgentUICoreMediator.ShowSearchableReportArea: 7851111b911c)
* در صورت فراخوانی متد نمایش گزارش جستجو محوری که از جنس نمای جدولی داده ها باشد (AgentUICoreMediator.ShowSearchableReportArea: 7851111b911c)
* در صورت کلیک بروی منوی داده از جنس نمایش داده های مرتبط در نمای باکسی (DataMenuArea.MenuUI\_MenuClicked: 28b138367084)
* در صورت کلیک بروی منوی داده از جنس نمایش داده های مرتبط در نمای جدولی (DataMenuArea.MenuUI\_MenuClicked: 28b138367084)

## منوی داده :

در واسط کاربری داده ها ممکن است در قالبها و فرمهای مختلفی نمایش داده شوند. ممکن است بصورت یک داده کامل یا نمایشی در یک فرم کار با داده و یا بصورت یک ردیف در نمایش جدولی و یا یک باکس گرافیکی ارائه شوند. فارغ از نوع نمایش قابلیتی در واسط کاربری بروی داده قابل تعریف است تا با نمایش منوهایی بروی هر داده، سایر فرمها و قابلیتهای سامانه برای آن داده در دسترس باشند. به منظور نمایش این منو از متد ShowMenuArea استفاده می شود (AgentUICoreMediator.ShowMenuArea: d9a11fdedec0). به این منو یک DataMenuAreaInitializer ارسال می شود که دارای ساختار زیر است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| SourceView | object | فرم مبدا |
| DataItem | DP\_DataView | داده مرتبط |
| HostDataViewArea | I\_DataArea | فرم نمای داده |
| HostDataViewItem | I\_DataViewItem | باکس داده مربوطه |
| DataMenuSettingID | int | شناسه ترکیب منوی داده |

خصوصیت HostDataViewArea در صورتی مورد استفاده است که منو از فرمهای نمای داده باز شده باشد. همچنین خصوصیت HostDataViewItem نیز در صورتی مقدار دارد که منو از یک باکس نمای داده باز شود. اساساً کاربردهای منوی داده یا همان متد ShowMenuArea موارد زیر می باشد:

* در فرم نمای باکسی داده ها
* در فرم نمای جدولی داده ها
* در فرم لینک داده
* در فرم گراف داده
* در فرم کارتابل
* در فرم گزارش جریان کار

DataMenuAreaInitializer ارسال شده در متد ShowMenuArea برای ایجاد یک DataMenuArea استفاده می شود که ابزار مدیریت منوها می باشد (DataMenuArea: 6f727758aef9). مهمترین بخش ابزار دریافت لیست منوها می باشد که توسط متد GetDataMenu انجام می شود (BizDataMenuSetting.GetDataMenu: dad2f7cbb995). انواع منو به دو دسته تقسیم می شوند، منوهای ثابت که تک موردی بوده و همواره برای داده نمایش داده می شوند و منوهای متغیر که چند موردی بوده و اینکه کدام موارد به عنوان منو نمایش داده شوند می بایست توسط کاربر تعیین شوند. لیست انواع منوهای ثابت به شرح زیر می باشد:

* منوی نمایش آرشیو داده
* منوی نمایش نامه های داده
* منوی نمایش گزارش جریان کارهای داده
* منوی نمایش فرم ورود اطلاعات داده

منوهای متغیر به علت تعدد موارد می بایست توسط کاربر تعیین شوند. برای مثال فرض کنید لیست روابط و گزارش های مربوطه یک داده ممکن است بیش از 30 مورد باشد. اینکه کدام روابط برای نمایش گزارشها در منو انتخاب شوند یک موضوع کلیدی در تعریف منو ها می باشد. لیست انواع منوهای متغیر به شرح زیر می باشد:

* منوی نمایش نمای جدول داده های یک رشته رابطه مشخص
* منوی نمایش نمای باکسی داده های یک رشته رابطه مشخص
* منوی نمایش گزارش داده های یک رشته رابطه مشخص
* منوی گزارش مستقیم داده

اساساً ترکیب منوهای متغیر در قالب موجودیتی به نام DataMenuSettingDTO تعریف می شود. برای یک موجودیت می توان بیشمار از این ترکیبها تعریف نمود. همچنین می توان یکی از این ترکیبها را هم به عنوان ترکیب منوی پیش فرض یک موجودیت تعیین نمود. حال هر جا که بحث نمایش داده و منوها آن مطرح است می توان یا ترکیب منوی خاصی را نیز تعیین نمود و یا در صورت عدم تعیین از ترکیب منوی پیش فرض استفاده نمود. اساساً برای دریافت منوها از متد GetDataMenu نیز دو متغیر اصلی ارسال می شود. یکی داده منبع و دیگری شناسه ترکیب منوی مورد نظر که همان خصوصیت DataMenuSettingID در DataMenuAreaInitializer می باشد. امّا اینکه آیا شناسه ترکیب منو به هنگام درخواست منوها مشخص است یا خیر بستگی به شرایط فرم میزبان داده ها دارد. برای مثال در صورتی که میزبان داده ها فرم نمای باکسی باشد در صورتی که این فرم بصورت عمومی و از منوهای فرم اصلی باز شده باشد طبیعتاً ترکیب منوی خاصی مشخص نشده و نهایتاً در دریافت منوها از ترکیب پیش فرض استفاده می شود. امّا اگر فرم نمای باکسی خود از یک گزارش باز شده باشد در آن گزارش ترکیب منو قابل تعیین شدن است و منوها طبق همان شناسه ترکیب دریافت و نمایش داده می شوند.

برای تعریف ترکیب منو ها از فرم مخصوصی در برنامه مدیریت فراداده استفاده می شود (frmDataMenuSetting: 3ebf70a010fe).

ساختار آن در زیر اشاره شده است:

|  |  |  |
| --- | --- | --- |
| خصوصیت | نوع | توضیحات |
| ID | object | شناسه |
| Name | DP\_DataView | نام ترکیب منو |
| EntityID | I\_DataArea | شناسه موجودیت |
| DataViewRelationships | List<DataMenuDataViewRelationshipDTO> | روابط نمای باکسی |
| GridViewRelationships | List<DataMenuSearchableReportRelationshipDTO> | روابط نمای جدولی |
| SearchableReportRelationships | List<DataMenuGridViewRelationshipDTO> | روابط گزارشات |
| DataItemReports | List<DataMenuDataItemReportDTO> | گزارشهای مستقیم |
| TargetDataMenuSettingID | int | شناسه تنظیمات منوی هدف |
| EntityListViewID | int | شناسه لیست نمایش |
| IconContent | Byte[] | آیکون منو |
| RelationshipID | int | شناسه انتقال رابطه برای نماها |

خصوصیت DataViewRelationships:

هر داده بنابر تعیین یک رشته رابطه به لیستی از داده های مرتبط منتهی می شود. حال می توان برای این داده های مرتبط یک نمای باکسی تعریف نمود.

امکان تعریف فرمول بروی ستون :

* + استفاده از کد تابع و ارتباط با خصوصیات موجودیت: محاسبه تعداد ساعت کل بروی ServiceItem با استفاده از کد فانکشن "محاسبه ساعات کار" و قراردادن در ستون HoursSpent
  + استفاده از توابع پایگاه داده : تعریف فرمول " محاسبه مبلغ " بروی ServiceItem که مبلغ را با ارسال دو فیلد HoursSpent و ServiceTypeEnum به StoredProcedure به نام sp\_CalculateServiceItemPrice محاسبه میکند.
  + استفاده از کد تابع به تنهایی:

1. محاسبه تاریخ روز و قرار دادن در ستون UpdateDate موجودیت ServiceItem با استفاده از کد تابع GetNow
2. محاسبه تاریخ روز و قرار دادن در ستون UpdateTime موجودیت ServiceItem با استفاده از کد تابع GetNow
   * استفاده از فرمول خطی :
3. تعریف فرمول خطی به نام " محاسبه هزينه" بروی ServiceConclusionItem که مبلغ را ServiceItem یا ServiceAdditionalItem می خواند. نتیجه این فرمول در ستون Price قرار میگیرد.
4. تعریف فرمول خطی به نام "جمع موارد" بروی ServiceConclusion که مبالغ ServiceConclusionItem مرتبط را جمع میکند و نهایتاً در ستون TotalPrice قرار میدهد.

* محاسبه فرمول بصورت پیش فرض :

محاسبه تاریخ روز شمسی و قرار دادن در ستون PersianDate موجودیت ServiceRequest با استفاده فرمول خطی. همچنین خصوصیت CalculateFormulaAsDefault این ستون باید مقدار True قرار داده شود.

محاسبه شناسه پست سازمانی و شناسه دفتر در در خواست سرویس بصورت پیش فرض

امکان استفاده از StoredProcedure :

امکان استفاده از DB Function :

امکان تعریف نماهای مختلف داده ListView:

لیست نمایش داده برای خلاصه سرویس

امکان تعریف نماهای جستجوی داده : SearchView:

لیست نمایش داده برای خلاصه سرویس

انواع :

به حالت Value/Text : فیلد EmployeeRole در جدول Employee ، در این حالت مقادیر از ستون EmployeeRole به حالت عددی خوانده میشوند اما در فرم مقادیر Text معادل نمایش داده میشوند.

به حالت Value تنها: فیلد WorkshopLevel در جدول Office، در این حالت لیست مقادیر از ستون WorkshopLevel عیناً خوانده و نمایش داده میشوند. حال میتواند این ستون عددی یا رشته ای باشد.

انواع و کاربردها

امکان استفاده از فیلد محاسباتی پایگاه داده : فیلد Name در موجودیت GenericPerson

امکان تعریف وضعیتها:

مقداردهی پیش فرض خصوصیات: خصوصیت کد در سازمان : که اگر isAgency مقدار true داشته باشد مقدار کد برابر 10 و اگر isWorkshop تیک بخورد مقدار پیش فرض کد 20 می شود.

فقط خواندنی سازی موجودیت : موجودیت شهر/استان که اگر نوع داده لود شده کشور باشد فرم کلاً فقط خواندنی می شود.( خود کمبو نوع نمی شود. بررسی شود)

غیرفعال سازی خصوصیت: مخفی کردن خصوصیت WorkshopLevel در واحد اداری وقتی تیک آژانس انتخاب می شود.

فقط خواندنی سازی خصوصیت: خصوصیت نوع در موارد خلاصه سرویس که اگر IsConfimed برابر با true باشد این خصوصیت فقط خواندنی می شود

غیرفعال سازی رابطه: مخفی کردن رابطه پدر در شهر/استان وقتی نوع کشور انتخاب می شود.

فقط خواندنی سازی رابطه: در موارد خلاصه سرویس اگر خصوصیت IsConfimed انتخاب شود روابط ServiceItem و ServiceAdditionalItem فقط خواندنی می شوند

بازه داده برای خصوصیت : فیلتر نقش 2 کارمند بر اساس نقش 1 در فرم کارمند

امکان تعریف BackednAction :

امکان اعمال کد تابع عمومی قبل از لود شدن داده موجود :

عمومی : کد تابع EditPersianDateMonthDay که تاریخهای شمسی که ماه و یا روز تک رقمی دارند را به ماه و روز دو رقمی اصلاح می کند.

موجودیت اختصاصی: کد تابع "اصلاح کد ملی" به روی موجودیت شخص حقیقی که در صورت نبودن دو صفر اول کد ملی به آنها دو صفر را اضافه می کند.

امکان اعمال کد تابع عمومی قبل از بروزرسانی شدن داده:

اختصاصی : کد تابع حذف عبارت "شرکت" از نام اشخاص حقوقی (در حال حاضر در صورتی که این فیلد تغییری کرده باشد اعمال می شود، بررسی شود که منطقی است یا خیر)

کد تابع عمومی : کد تابع عمومی "اعتبارسنجی درخواست" که پیغام خطای کاربر غیر مجاز می دهد در صورتی که شناسه درخواست دهنده برابر با عدد 666 باشد

امکان اعمال کد تابع عمومی قبل از حذف داده:

اختصاصی : کد تابع "اعتبارسنجی حذف درخواست سرویس" که به هنگام حذف درخواست سرویس اگر خصوصیت userId با شناسه کاربر جاری یکسان نباشد خطا می دهد

کد تابع عمومی : کد تابع عمومی "اعتبارسنجی درخواست حذف" که پیغام خطای کاربر غیر مجاز می دهد در صورتی که شناسه درخواست دهنده برابر با عدد 666 باشد.

امکان اعمال کد تابع عمومی بعد از بروزرسانی داده :

اختصاصی : تابع دیتابیس sp\_CalculateCustomerValue که ارزش مشتری را با گرفتن پارامتر شناسه از ServiceConclusion محاسبه می کند

امکان اعمال کد تابع عمومی بعد از حذف داده :

اختصاصی : تابع دیتابیس sp\_CalculateCustomerValueByServiceRequestID که ارزش مشتری را با گرفتن پارامتر شناسه درخواست از ServiceConclusion محاسبه می کند. به این علت با بالایی تفاوت دارد زیرا بعد از حذف دیگر شناسه اصلی وجود ندارد که از تابع بالا استفاده شود.

امکان فیلتر کردن روابط :

فیلتر رابطه نوع سرویس در موجودیت مورد بررسی درخواست (ServiceRequestReviewItems) که بر اساس درخواست سرویس فیلتر می شود (انواع درخواست سرویس تعریف شده در درخواست سرویس مرتبط)

فیلتر رابطه ServiceItemRepair با ServiceActionType که بر اساس نوع سرویس از ServiceItem و نوع محصول از درخواست سرویس فیلتر می شود.

امکان تعریف رابطه : ایجاد رابطه یک به چند بین Product از دیتابیس DBProducts و ProductItem از دیتابیس DBProductService

امکان تعریف انواع اعتبارسنجی ها :

امکان فقط خواندنی نمودن موجودیتها :

امکان تعریف انواع اعتبارسنجی به روی ستونها :

امکان تعریف لیست مقادیر :

لیست مقادیر نقش کارمند در فرم کارمند

امکان فقط خواندنی نمودن ستونها : فقط خواندنی نمودن خصوصیت PersianDate در ServiceRequest

امکان ورود اطلاعات اجباری ستونها : اجباری نمودن ستون Name در شخص حقوقی

اعتبارسنجی ستونها بر حسب نوع:

ستونهای رشته ای :

امکان تعریف حداکثر طول رشته: فیلد کد در جدول Office

امکان تعریف حداقل طول رشته: فیلد کد در جدول Office

امکان تعریف فرمت: با تعریف فرمت میتوان نسبت به اعتبارسنجی ستونهای رشته ای به هنگام ثبت فرم اقدام کرد.

مثال : فیلد EmailAddress در جدول شخص عمومی (باید فرمت با مقدار ^(?!\.)(""([^""\r\\]|\\[""\r\\])\*""|([-a-z0-9!#$%&'\*+/=?^\_`{|}~]|(?<!\.)\.)\*)(?<!\.)@[a-z0-9][\w\.-]\*[a-z0-9]\.[a-z][a-z\.]\*[a-z]$ برای این ستون ست شود)

ستونهای عددی :

امکان تعریف تعداد اعشار و عدد صحیح : فیلد UserRate در خلاصه سرویس که یک رقم صحیح و یک رقم اعشار دارد

امکان تعریف بیشینه و کمینه : فیلد UserRate در خلاصه سرویس که مقدار بیشینه آن 5 و کمینه 1 می باشد

ستون تاریخ :

امکان بزگتر کو چکتر تاریخ روز و ...

اعتبارسنجی روابط :

فقط خواندنی نمودن رابطه : ؟؟

اجباری نمودن رابطه : اجباری نمودن رابطه با شهرها در GenericPersonAddress

اعتبارسنجی روابط بر حسب نوع:

روابط ارث بری :

ارث بری Disjoint باشد : رابطه ارث بری بین GenericPerson و LegalPerson و RealPerson

ارث بری Disjoint نباشد : ؟؟؟

ارث بری TolatParticipation باشد : رابطه ارث بری بین GenericPerson و LegalPerson و RealPerson

ارث بری TolatParticipation نباشد : رابطه ارث بری بین GenericPerson و Customer

روابط اتحاد :

اتحاد TolatParticipation باشد : ؟؟؟

اتحاد TolatParticipation نباشد : رابطه ارث بری بین ServiceConclusionItem و ServiceItem و ServiceAdditionalItem

اعتبارسنجی اختصاصی موجودیتها :

امکان دو نوع اعتبار سنجی وجود دارد :

1. اعتبارسنجی اختصاصی بر اساس فرمول اعتبارسنجی : فرمول اعتبارسنجی برای درخواست سرویس که با استفاده از کد تابع اعتبارسنجی درخواست سرویس اجازه ثبت درخواست برای کالا با عنوان برند "Hisenese" را نمی دهد.
2. کد تابع قبل از بروز رسانی : به " امکان اعمال کد تابع عمومی قبل از بروزرسانی شدن داده" مراجعه شود.

امکان تعریف دکمه های Command :

امکان تعریف DataView و GridView :

جستجوی عمومی نمای گرافیکی داده (DataView) : منوی عمومی جستجو و نمایش گرافیکی داده که امکان انتخاب نوع موجودیت و سپس جستجوی پیش فرض را بروی آن موجودیت می دهد.

جستجوی عمومی نمای جدولی داده (GridView) : منوی عمومی جستجو و نمایش جدولی داده که امکان انتخاب نوع موجودیت و سپس جستجوی پیش فرض را بروی آن موجودیت می دهد.

امکان تعریف نمای داده موجودیت در منوی برنامه : برای هر موجودیت میتوان نمای داده آن را (با تنظیمات منوی داده پیش فرض) در منو تعریف نمود. نمونه : نمای داده مشتری در زیر منوی نمای داده.

امکان تعریف گرید داده موجودیت در منوی برنامه : برای هر موجودیت میتوان گرید داده آن را (با تنظیمات منوی داده پیش فرض) در منو تعریف نمود. نمونه : گرید داده مشتری در زیر منوی نمای داده.

امکان تعریف گزارش نمای داده : به بخش گزارشات مراجعه شود.

امکان تعریف گزارش گرید داده : به بخش گزارشات مراجعه شود.

تنظیمات منوی داده :

امکان تعریف رشته روابط گرید داده یا نمای داده و اختصاصی سازی منوی داده آنها : برای موجودیت مشتری نمای داده رشته رابطه با خلاصه درخواست سرویس (از طریق درخواست سرویس و سپس خلاصه سرویس) تعریف شده است. تنظیمات منوی خلاصه سرویس نیز برای این رابطه تعیین شده است که خود دارای گرید داده رابطه یا موارد خلاصه سرویس می باشد.

امکان تعریف رشته روابط گزارشات : برای موجودیت مشتری گزارشاتی بر اساس رشته رابطه با خلاصه درخواست سرویس (از طریق درخواست سرویس و سپس خلاصه سرویس) تعریف شده است. رابطه با گزارش لیست صورتحساب، گزارش دایره ای صورتحساب و گزارش نمای داده صورتحساب

دسترسی به منوی داده : چه در نمای داده، گرید داده و یا قلم اطلاعاتی داده چه در فرم ورود اطلاعات و چه کارتابل و .. می توان به منوی داده دسترسی داشت. سپس تنظیمات داده اختصاصی و یا پیش فرض به روی منوی داده اعمال می شود.

امکان استفاده از View :

نمای view\_ServieRequests در سامانه بروزرسانی شده است و گزارش چارت ساعات سرویس هر برند برای این نما تعریف گردیده است. همچنین ارتباطی بین این نما و ServiceRequest تعریف شده و از خصوصیت HoursSpent این نما در ستونهای نمایش پیش فرض ServiceRequest استفاده شده است. همچنین تنظیمات منوی داده برای این نما به تنظیمات منوی داده درخواست سرویس ارجاع شده است.

امکان تعریف انواع گزارشات:

انواع گزارشات :

گزارشات خارجی مستقیم به روی قلم داده :

گزارشات جستجو محور :

امکان تعریف جستجوی پیش فرض : ساختار SearchRepository

برای این دسته از گزارشات علاوه بر خصوصیات و قابلیتهای اختصاصی هر نوع گزارش، امکان تعریف جستجوهای پیش فرض نیز به منظور فیلتر نمودن بازه داده ها مقدور است.

گزارش نمای داده : گزارش نمای داده صورتحساب در بخش گزارشات. در این گزارش از جستجوی پیش فرض به نام "جستجوی صورتحساب از ابتدای سال" استفاده شده است. همچنین تنظیمات منو همان "تنظیمات منوی صورتحساب و جزئیات" می باشد.

گزارش گرید داده : گزارش گرید داده صورتحساب در بخش گزارشات. در این گزارش از جستجوی پیش فرض به نام "جستجوی صورتحساب از ابتدای سال" استفاده شده است. همچنین تنظیمات منو همان "تنظیمات منوی صورتحساب و جزئیات" می باشد.

گزارش چاپی لیستی ساده : گزارش چاپی لیست صورتحساب ساده در بخش گزارشات. در این گزارش از جستجوی پیش فرض به نام "جستجوی صورتحساب از ابتدای سال" استفاده شده است. همچنین لیست نمایش به نام "لیست پیش فرض خلاصه سرویس" می باشد.

گزارش چاپی لیستی با گروهبندی : در بخش گزارشات به نام "گزارش چاپی لیست صورتحساب گروهبندی" می باشد. در این گزارش گروهبندی بر اساس ستون نمایشی عنوان برند می باشد.

گزارش چاپی لیستی با زیر گزارش : در بخش گزارشات به نام "گزارش چاپی لیست صورتحساب زیر گزارش" می باشد. زیر گزارش مرتبط نیز همان موارد صورتحساب به نام "گزارش چاپی لیست موارد صورتحساب" است که ارتباط گزارش و زیر گزارش از طریق ستون “ID” از سمت گزارش و ستون " ServiceConclusionID" از سمت زیر گزارش می باشد.

گزارش چارت ستونی : گزارش با عنوان "گزارش چارت ستونی صورتحساب" که دسته بندی اول آن بر اساس نام برند و دسته بندی دوم بر اساس نامه دفتر می باشد. به این ترتیب بر اساس این دو دسته نمودار ستونی تعداد صورتحسابها نمایش داده می شوند.

گزارش چارت دایره ای : گزارش با عنوان "گزارش چارت دایره ای صورتحساب" که دسته بندی آن بر اساس نام برند می باشد که بر اساس آن نمودار دایره ای تعداد صورتحسابها نمایش داده می شوند.

گزارش چارت خطی : گزارش با عنوان "گزارش چارت خطی صورتحساب" که دسته بندی اول آن بر اساس نام برند و دسته بندی دوم بر اساس نامه دفتر می باشد. به این ترتیب بر اساس این دو دسته نمودار خطی تعداد صورتحسابها نمایش داده می شوند.

گزارش چارت راداری : گزارش با عنوان "گزارش چارت راداری صورتحساب" که دسته بندی اول آن بر اساس نام برند و دسته بندی دوم بر اساس نامه دفتر می باشد. به این ترتیب بر اساس این دو دسته نمودار راداری تعداد صورتحسابها نمایش داده می شوند.

گزارش کراس تب : گزارش با عنوان "گزارش کراس تب صورتحساب" که ستونهای آن بر اساس نام برند و مدل محصول و ردیفهای آن بر اساس نامه دفتر می باشد. به این ترتیب بر اساس این ستونها و ردیفها جدول کراس تب تعداد صورتحسابها نمایش داده می شوند.

گزارشات جستجویی خارجی : با ثبت تنظیمات گزارش خارجی برای هر موجودیت، یک جدول با پیشوند "xr\_" در پایگاه داده استقرار موجودیت ایجاد می شود. ستونهای این جدول کلیدهای اصلی موجودیت و یک ستون به نام ReportKey از نوع int می باشد.

بعد از هر بار جستجوی داده ها در فرم گزارش، کلیدهای اصلی داده ها به همراه یک شناسه ایجاد شده ReportKey در این جدول قرار می گیرند و سپس این ReportKey به عنوان پارامتر به آدرس تنظیم شده گزارش ارسال می شود. حال کد پشت گزارش میتواند برای بدست آوردن کلیدهای رکوردهای جستجو شده از این ReportKey استفاده نموده و داده ها را فیلتر نماید. برای مثال گزارشی به آدرس "<http://dolatkiam/ReportServer/Pages/ReportViewer.aspx?%2fRequestConclusionsByReportKey>" در ریپورتینگ سرویس برای صورتحسابها تعریف شده است. این گزارش از یک SP استفاده می کند که جدول ServiceConclusion را با xr\_ServiceConclusion جوین زده و از پارامتر ReportKey برای فیلتر کردن داده ها استفاده می کند. SP استفاده شده به شکل زیر می باشد.

USE [DBProductService]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[Report\_ServiceRequestConclusions] Script Date: 7/27/2021 7:00:52 PM \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[Report\_ServiceRequestConclusions]

-- Add the parameters for the stored procedure here

@ReportKey bigint

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

--EXEC sp\_CreateReportTable 'Report\_City',@ReportKey

SELECT \* from Accounting.ServiceConclusion

inner join dbo.xr\_ServiceConclusion as filter on Accounting.ServiceConclusion.ID=filter.id where filter.ReportKey=@ReportKey

END

بنابر این برای تعریف گزارشات خارجی بهتر است 3 مرحله زیر طی شود:

1. گزارش خارجی بصورت کلی و بدون استفاده از جدول واسط xr طراحی شود.
2. تنظیمات گزارش خارجی در برنامه معرفی شود. به این ترتیب جدول واسط xr نیز ساخته می شود.
3. گزارش خارجی اصلاح شود و در Query پشت آن جدول xr و پارامتر ReportKey نیز به منظور فیلتر داده ها اعمال شوند.

گزارشات مستقیم :

بروی مشتری گزارش مستقیمی تعریف شده که با استفاده از پارامترهای تعریف شده در تنظیمات آن، گزارش خارجی را به آدرس <http://dolatkiam/ReportServer/Pages/ReportViewer.aspx?%2fRequestConclusionsByCustomerID> فراخوانی می کند. این گزارش هم در منوی برنامه و هم در تنظیمات منوی مشتری قابل دسترسی است.

امکان تعریف DataLink :

تعریف گزارش لینک داده بنام گزارش لینک ریپورت که بین Region از دیتابیس خدمات و BrandProductType از دیتابیس محصولات که بر اساس رشته رابطه شهر/استان (Region )، آدرس شخص، شخص، مشتری، درخواست سرویس، قلم کالا، محصول و برند/نوع محصول (BrandProductType) می باشد. در این رشته رابطه برای موجودیتهای مشتری و درخواست سرویس از تنظیمات منوی اختصاصی استفاده شده است.

امکان تعریف گراف داده :

تعریف گزارش گراف داده بنام" گزارش گراف" که از مشتری شروع شده و رابطه با درخواست سرویس و صورتحساب شرویس را نمایش می دهد. در این رشته رابطه برای موجودیتهای مشتری و درخواست سرویس از تنظیمات منوی اختصاصی استفاده شده است.

امکان آرشیو :

امکان تعریف پوشه اختصاصی برای نوع موجودیت : پوشه ضمانتنامه برای درخواست سرویس

امکان تعریف پوشه های عمومی: تعریف پوشه مدارک عمومی

امکان تعریف تگ های اختصاصی برای نوع موجودیت : تگ مدرک اصلی برای درخواست سرویس

امکان تعریف پوشه های عمومی: تعریف تگ اصل و کپی به صورت عمومی

امکان تعریف نمایش آرشیو داده مرتبط : تعریف رابطه آرشیو بررسی درخواست سرویس به روی موجودیت درخواست سرویس و بالعکس

نمایش خودکار آرشیو برای داده های مرتبط : این خصوصیت برای بررسی درخواست سرویس فعال است و موارد مرتبط بصورت خودکار نمایش داده می شوند.

دسترسی آرشیو : کاربر کوثری اجازه افزودن یا حذف موارد آرشیو را روی درخواست سرویس ندارد اما روی بررسی درخواست سرویس دارد.

امکان نامه ها:

امکان تعریف نمایش نامه های داده مرتبط : تعریف رابطه نامه بررسی درخواست سرویس به روی موجودیت درخواست سرویس و بالعکس

نمایش خودکار نامه ها برای داده های مرتبط : این خصوصیت برای بررسی درخواست سرویس فعال است و موارد مرتبط بصورت خودکار نمایش داده می شوند.

تعریف نوع نامه عمومی : نوع نامه "نوع نامشخص" به صورت عمومی

تعریف نوع نامه اختصاصی : نوع نامه "درخواست سرویس" برای موجودیت درخواست سرویس

دسترسی نامه : کاربر کوثری اجازه افزودن یا حذف موارد نامه را روی درخواست سرویس ندارد اما روی بررسی درخواست سرویس دارد.

انواع ایجاد نامه :

نامه داخلی با انتخاب فایل : ایجاد نامه با عنوان "نامه داخلی موجود " برای اولین داده درخواست سرویس، فایل مرتبط با این نامه از آدرس D:\Files\ServiceRequestInternalLetter.docx بارگذاری می شود.

نامه داخلی تولید فایل با فیلدهای ساده : ایجاد نامه با عنوان " نامه توليد شده نمونه" برای اولین داده درخواست سرویس، قالب مرتبط با این نامه از آدرس D:\Files\ ServiceRequestLetterTemplate.docx بارگذاری می شود.

نامه داخلی تولید فایل با فیلدهای رابطه ای : : ایجاد نامه با عنوان "نامه توليد شده پيچيده" برای اولین داده بررسی درخواست سرویس، قالب مرتبط با این نامه از آدرس D:\Files\ ServiceRequestReviewLetterTemplate.docx بارگذاری می شود. در این قالب در فیلد رابطه ای Rel\_SrvReqType\_Start تمامی انواع درخواست مربوط به درخواست سرویس مرتبط با بررسی درخواست سرویس نمایش داده می شود. سپس در فیلد رابطه ای Rel\_SrvReqRevItem\_Start موارد خود بررسی درخواست سرویس نمایش داده می شود. درون این رابطه از رابطه دیگری که ارتباط موارد را با نوع درخواست تعیین میکند به نام Rel\_ReqType\_Start استفاده شده است.

ایجاد نامه با استفاده از منبع خارجی : در تنظیمات نامه کد منبع خارجی مشخص می شود و به این ترتیب می توان نامه با استفاده از منبع خارجی ایجاد کرد. برای اولین داده درخواست سرویس تولید می شود.

تولید و تکرار رابطه در جدول را هم داریم، بعداً تست شود.

تنظیمات نامه :

کد فانکشن قبل از لود نامه : فانکشن BeforeLoad از MyTestImplLibrary

کد فانکشن قبل از ذخیره نامه : فانکشن BeforeSave از MyTestImplLibrary

کد فانکشن بعد از ذخیره نامه : فانکشن AfterSave از MyTestImplLibrary

کد فانکشن دریافت اطلاعات نامه از منبع خارجی : فانکشن ExternalCode از MyTestImplLibrary

کد فانکشن ارسال نامه داخلی به خارجی : فانکشن ConvertToExternal از MyTestImplLibrary

دسترسی داده مستقیم:

دسترسی داده در دو نوع انجام می شود. دسترسی دریافت/نمایش اطلاعات و دسترسی فقط خواندنی نمودن اطلاعات.

برای موجودیت درخواست هزینه دسترسی داده در دو حالت تعریف شده است. اینکه کاربر راهبر کل باشد و یا شناسه دفتر برابر با شناسه سازمان خارجی کاربر باشد.

دسترسی اصلاح داده: برای موجودیت درخواست هزینه دسترسی اصلاح داده در دو حالت تعریف شده است. اینکه کاربر راهبر کل باشد و یا شناسه پست سازمانی درخواست برابر با شناسه پست سازمانی کاربر باشد.(یا به عبارتی خود کاربر درخواست را ایجاد کرده باشد)

دسترسی داده غیر مستقیم: برای موجودیت خلاصه سرویس دسترسی داده از دخواست سرویس مربوطه خوانده و اعمال می شود.

لاگ داده :

تعریف جریان کار:

امکان تعریف وضعیت ها در یک جریان کار : برای جریان کار "درخواست سرويس" 7 وضعیت نمونه تعریف شده است. امکان تعریف فرمول برای هر وضعیت : برای هر وضعیت فرمول هایی می توان تعریف نمود که در صورت تخقق فرمول برای داده مرتبط امکان ایجاد چنین وضعیتی در جریان کار فراهم باشد.

امکان تعریف فعالیت برای هر وضعیت : برای هر وضعیت می توان لیست فعالیتهایی تعریف نمود که در صورت محقق شدن وضعیت آن فعالیتهای انجام شوند.

|  |  |  |
| --- | --- | --- |
| عنوان وضعیت | نوع وضعیت | فرمول |
| ثبت درخواست | وضعیت شروع | HasReview برقرار نباشد |
| بررسي درخواست | وضعیت در جریان |  |
| در حال سرويس مي باشد | وضعیت در جریان |  |
| در حال صدور صورتحساب مي باشد | وضعیت در جریان |  |
| صورتحساب تسويه شد و اتمام کار | وضعیت پایان | فرمول Cleared برقرار باشد |
| درخواست لغو شد و اتمام کار | وضعیت پایان |  |
| قابل تعمير نمي باشد و اتمام | وضعیت پایان |  |

امکان تعریف تبدیل های بین وضعیت و اقدامات و فعالیتهای مرتبط : برای هر انتقال بین وضعیت مجموعه ای از مفاهیم را میتوان تعریف نمود که در زیر به آنها اشاره شده است.

اقدامات : اقدامات مجموعه وظایفی هستند که در صورت انجام شدن عملیات انتقال وضعیت انجام می شود. هر اقدام خصوصیات زیر را دارد :

کاربران هدف : مجموع ای از نقشها که توانایی انجام آن اقدام را در جریان کار محوله دارند.

فرمول : فرمولهایی که در صورت محقق بودن یا نبودن امکان اعتبارسنجی انجام شده بودن اقدام را می دهد.

گروه فرمها : مجموعه ای قابل تعریف از فرمها که در محدوده کل جریان کار تعریف می شود و با متصل کردن آنها به هر اقدام امکان ورود اطلاعات آن فرمها را همراه با اقدامات متناسب در کارتابل به کاربر می دهد.

امکان انجام اقدام توسط چند کاربر : به هنگام ارجاع کار می توان همزمان چند کاربر هدف را جهت انجام اقدام مشخص نمود.

فعالیتای هر انتقال : برای هر انتقال می توان لیست فعالیتهایی تعریف نمود که در صورت محقق شدن آن انتقال وضعیت فعالیتهای انجام شوند.

امکان تعریف وضعیتهای وتبدیلها و اقدامات مرتبط به صورت نموداری : تمامی وضعیتها، تبدیلها و اقدامات مرتبط را می توان به صورت نموداری تعریف نمود.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| از وضعیت | به وضعیت | نام تبدیل | اقدامات مرتبط | فرمول اقدامات | نقشهای اقدامات | گروه فرمها |
| ثبت درخواست | بررسي درخواست | ارسال به بررسي درخواست | تایید ثبت درخواست |  | راهبر اداره  راهبر کل  اپراتور ثبت درخواست | ثبت درخواست شامل فرم ثبت درخواست |
| بررسي درخواست | در حال سرويس مي باشد | ارسال به تعميرات | تایید ارسال به تعميرات | HasReview برقرار باشد | راهبر اداره  راهبر کل  بررسی کننده درخواست | بررسی درخواست شامل فرم بررسی درخواست |
| در حال سرويس مي باشد | در حال صدور صورتحساب مي باشد | ارسال به صورتحساب | تاييد و ارسال به صورتحساب |  | راهبر اداره  راهبر کل  تعمیرکار | ثبت تعميرات شامل فرم مورد سرویس |
| در حال صدور صورتحساب مي باشد | صورتحساب تسويه شد و اتمام کار | ارسال به تسويه و اتمام | اتمام جريان کار |  | راهبر اداره  راهبر کل  مسئول مالی سرویس | ثبت اطلاعات صورتحساب شامل فرم خلاصه سرویس |
| بررسي درخواست | ثبت درخواست | اطلاعات ثبتي ناقص مي باشد | بازگشت به مرحله ثبت سرويس |  | راهبر اداره  راهبر کل  بررسی کننده درخواست | بررسی درخواست شامل فرم بررسی درخواست |
| در حال سرويس مي باشد | بررسي درخواست | قابل سرويس نمي باشد | بازگشت به بررسي سرويس |  | راهبر اداره  راهبر کل  تعمیرکار |  |
| بررسي درخواست | درخواست لغو شد و اتمام کار | درخواست قابل اجرا نمي باشد | درخواست قابل اجرا نمي باشد |  | راهبر اداره  راهبر کل  بررسی کننده درخواست |  |

کارتابل و تعریف جریان کار و ارجاع جریان کار:

توضیحات بعدا اضافه شود

تعریف کاربران و سازمانها :

امکان تعریف نوع سازمان : نوع سازمان تعمیرگاه، دفتر خدمات

امکان تعریف سازمانها : تعریف مرکز ونک و تعریف تعمیرگاه تجریش

امکان تعریف نوع نقش : نقش اپراتور ثبت درخواست، تعمیرکار، بررسی کننده درخواست، راهبر اداره، مسئول مالی سرویس و راهبر کل

امکان تعریف ارتباط بین نوع سازمان و نوع نقش :

نقشهای نوع سازمان تعمیرگاه شامل : تعمیرکار و راهبر اداره

تقشهای دفتر خدمات شامل: اپراتور ثبت درخواست، بررسی کننده درخواست، راهبر اداره، مسئول مالی سرویس ، راهبر اداره و راهبر کل

امکان تعریف پست سازمانی و کاربران مرتبط :

|  |  |  |
| --- | --- | --- |
| نوع نقش و سازمان | نام پست | کاربر مرتبط |
| نقش تعمیرکار در سازمان تعمیرگاه تجریش | تعمیرکار 1 | Rezayi |
| نقش تعمیرکار در سازمان تعمیرگاه تجریش | تعمیرکار 2 | Zahed |
| نقش تعمیرکار در سازمان تعمیرگاه تجریش | تعمیرکار 3 | Pirazad |
| نقش راهبر اداره در سازمان تعمیرگاه تجریش | راهبر 1 | Rezayi |
| نقش اپراتور ثبت درخواست در مرکز ونک | اپراتور 1 | Dolatkhah |
| نقش اپراتور ثبت درخواست در مرکز ونک | اپراتور 2 | Movaseghi |
| نقش بررسی کننده درخواست در مرکز ونک | بررسی کننده سرویس 1 | Kosari |
| نقش راهبر اداره در مرکز ونک | راهبر 1 | Afshari |
| نقش مسئول مالی سرویس در مرکز ونک | مسئول مالی سرویس 1 | Karimi |
| نقش راهبر کل در مرکز ونک | راهبر کل 1 | Dolatkia |

امکان تعریف دسترسی ها :

امکان تعریف دسترسی برای کاربران در سطوح زیر :

دسترسی نوع سازمان، دسترسی نوع نقش، دسترسی ارتباط نقش و نوع سازمان، دسترسی سازمان و دسترسی پست سازمانی

امکان تعریف دسترسی به روی منابع زیر :

دیتابیس، شمای دیتابیس، موجودیت، رابطه، ستون، گزارش، دکمه دستوری، زیر سیستم

امکان تعریف انواع دسترسی : عدم دسترسی، فقط خواندنی، اصلاح، اصلاح و حذف

درخت منابع و نوع دسترسی.

بنابراین دسترسی به منابع را میتوان از بین هر کدام از سطوح کاربری و نوع منبع ذکر شده انتخاب کرد. سعی می شود در مثالهای زیر هر کدام از سطوح و یا منابع ذکر شده بکار گرفته شوند :

دسترسی نوع سازمان :

نوع سازمان تعمیرگاه به کل دیتابیس دسترسی فقط خواندنی، نمایش آرشیو و نمایش نامه دارد.

نوع سازمان دفتر خدمات به کل دیتابیس دسترسی فقط خواندنی، نمایش آرشیو و نمایش نامه دارد.

دسترسی نوع نقش :

نقش تعمیرکار به شمای تعمیرگاه دسترسی اصلاح، حذف و اصلاح آرشیو و اصلاح نامه دارد.

نقش اپراتور ثبت درخواست: به موجودیتهای درخواست سرویس، انواع درخواست سرویس، مشتری، شخص، شخص حقیقی، شخص حقوقی ،آدرس و محصول دسترسی اصلاح ، حذف و اصلاح آرشیو و اصلاح نامه دارد.

نقش بررسی کننده درخواست سرویس به موجودیتهای بررسی درخواست سرویس و مورد درخواست سرویس دسترسی اصلاح ، حذف و اصلاح آرشیو و اصلاح نامه دارد.

نقش مسئول مالی به شمای Accounting دسترسی اصلاح ، حذف و اصلاح آرشیو و اصلاح نامه دارد.

نقش راهبر اداره به شمای Common دسترسی اصلاح ، حذف و اصلاح آرشیو و اصلاح نامه دارد. این نقش به موجودیت شهر/استان (که در شمای Common می باشد) دسترسی فقط خواندنی دارد.

نقش راهبر کل به کل پایگاه داده دسترسی اصلاح و حذف دارد.

دسترسی نوع نقش نوع سازمان :

نقش راهبر اداره در نوع سازمان تعمیرگاه به موجودیتهای نوع سرویس و قلم کالا دسترسی فقط خواندنی دارد. همچنین نقش راهبر اداره در نوع سازمان تعمیرگاه به ستون کدمشتری در جدول مشتری دسترسی ندارد.

دسترسی سازمان :

به نظرم مفهوم دسترسی سازمان حذف شود