**一、理解DataAccess中enum和数据库存储值之间的相互转换逻辑：**

1. DataAccess组件会先检查是否有自定义enum和数据库存储值之间的映射关系，如果有，则会按照自定义映射关系进行转换；

2. 如果没有自定义enum和数据库存储值之间的映射关系，那么对于SQL参数赋值，DataAccess会把enum先转换为其对应的int数值，然后再赋值给SQL参数；对于是从DB中读取的数据转换为enum，则会直接使用Enum.Parse方法来进行转换（有可能会抛出转换失败的异常）；

**二、在DataAccess中枚举映射规则的使用指导说明：**

1. 当enum的数据库存储值是int，并且正好和其enum的int值是一致的，那么不用做任何定义和特别处理，DataAccess将自然支持enum和数据库存储值之间的双向转换；

2. 当enum的数据库存储值是char等字符或字符串时，亦或者enum的数据库存储值是int但其值和enum的int值不一致时，那么需要使用ECCentral.Service.Utility.DataAccess工程中的EnumCodeMapper类型，来自定义enum和数据库存储值之间的映射关系，DataAccess组件则会按照该mapper所定义的映射关系，来自动做enum和数据库存储值之间的双向转换；自定义映射关系的具体做法是在每个Domain的DataAccess工程下都有一个EnumCodeMapRegister类型，用来加入映射代码，该类型实现了IStartup接口并会被配置在WebHost的Configuration/Autorun.config中，以保证系统启动时就会执行映射关系的建立；

**三、在DataAccess中支持自动完成enum和数据库存储值之间的相互转换的方法：**

1. DataCommand.InnerAddInputParameter： 如果传入的object的GetType()返回的类型是enum，那么将使用上面第一大点的转换逻辑实现enum到对应的DB存储值的自动转换；

2. DataCommand.ReplaceParameterValue：如果传入的object的GetType()返回的类型是enum，那么将使用上面第一大点的转换逻辑实现enum到对应的DB存储值的自动转换；

3. DataCommand.SetParameterValue：如果传入的object的GetType()返回的类型是enum，那么将使用上面第一大点的转换逻辑实现enum到对应的DB存储值的自动转换；

4. DataCommand.SetParameterValue<T>及相关重载：如果传入的T类型实体对象的属性上有enum的，那么将使用上面第一大点的转换逻辑实现enum到对应的DB存储值的自动转换；

5. DataCommand.ExecuteEntity<T>及相关重载：如果将返回的T类型实体对象的属性上有enum的，那么将使用上面第一大点的转换逻辑实现DB中读取值到enum的自动转换；

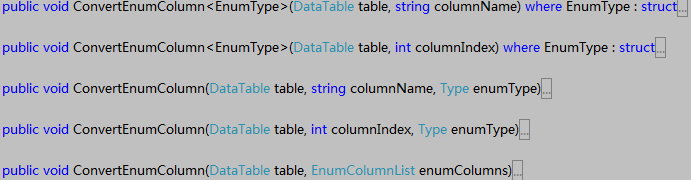
6. DataCommand.ExecuteEntityList<T, C>及相关重载：如果将返回的T类型实体对象的属性上有enum的，那么将使用上面第一大点的转换逻辑实现DB中读取值到enum的自动转换；

7. DataCommand.ExecuteEntityList<T>及相关重载：如果将返回的T类型实体对象的属性上有enum的，那么将使用上面第一大点的转换逻辑实现DB中读取值到enum的自动转换；

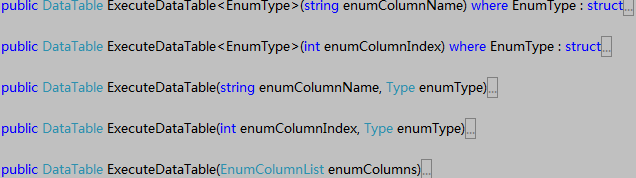
8. DynamicQuerySqlBuilder中涉及到的SQL入参赋值或拼接SQL的方法，也都支持enum到对应的DB存储值的自动转换

**四、对于No Biz Query时，返回的DataTable中，涉及到枚举相关字段的处理：**

1. 在DataCommand类中提供了ConvertEnumColumn方法，可以传入一个DataTable，并通过方法参数指明DataTable中哪些列是enum相关的列以及指明其具体的enum类型，然后由方法内部自动来将DataTable中的相应数据进行转换处理；



2. 在DataCommand类中也直接提供了相应的ExecuteDataTable的重载，用来直接返回已经将枚举相关列数据转换完毕的DataTable，其方法内部使用的就是第1小点里的ConvertEnumColumn相关方法；

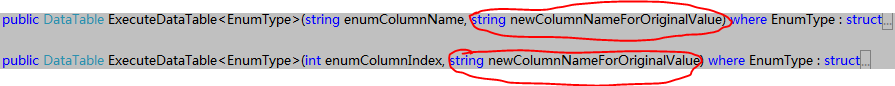


通过上面的2中方式，都可以保证Enum在DataAccess层就被处理完毕，这样返回给上层调用者的，都将是enum了，上层看到的也仅仅是enum，而不用知道、也不用去关心DB中具体存储的值是多少了。

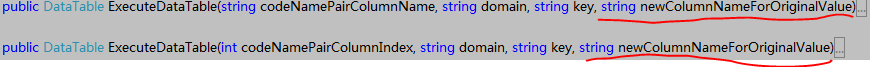
**附1：对于在No Biz Query中，CodeNamePair相关字段的处理，和enum的处理类似，也是在DataAccess层提供相关的方法（使用：1. ConvertCodeNamePairColumn方法； 2. ExecuteDataTable相关重载），将DB中存储的code字符，直接转换为显示用的name，然后再返回给上层调用者，这样No Biz Query的上层调用者拿到的就是用于显示的文本了；**

**------------------------------------------------**

**附2：对于一些No Biz Query的功能，可能除了需要CodeNamePair的显示文本外，界面上还需要使用到Code的值（比如判断控制一些功能按钮的显示等），所以ConvertCodeNamePairColumn、 ConvertEnumColumn以及ExecuteTable方法都提供了相应的重载，可以为DB中原始读出的值的列指定一个新的列名，来保证返回的结果集中会包含有DB原始值的列，以供使用；**



、



**方法ConvertCodeNamePairColumn、 方法ConvertEnumColumn以及EnumColumnList和CodeNamePairColumnList的Add方法也都有增加了**string newColumnNameForOriginalValue**入参的重载方法，这里就不详细展示了，请直接参照代码。**

**------------------------------------------------**

**附3：对于一些枚举字段可能存在历史遗留数据的问题，比如：对于枚举XXX的Void枚举项，在09年以前在DB里存储使用的字符’A’来表示，而09年改版后至今，都已经在DB里存储使用字符’X’来表示了枚举XXX的Void枚举项了，所以我们现在会设计字符’X’映射到枚举XXX的Void枚举项，那么当我们从DB里Query出09年以前的数据出来，转化到我们的Entity里时就会出现问题，因为‘A’字符无法转化为枚举XXX的Void枚举项了；**

**针对这种情况，我们的处理方法是：**

在枚举XXX所属的Domain的SqlDataAccess工程的EnumCodeMapRegister里，调用EnumCodeMapper.AddExtraCodeMap<TEnum>方法，为枚举XXX注册额外映射的Code，这个方法所注册的enum映射到的额外的Code，只会在Code ---》 Enum的时候起作用，而不会影响到Enum---》Code的转化过程，Enum---》Code的映射仍然还是按照EnumCodeMapper.AddMap<TEnum>方法所注册的信息来处理；