ABP VNext 部门规范参考

一、实体定义

```
    [Table("xxx")]
    public class TestPaper : FullAuditedAggregateRoot<Guid>, IMul
tiTenant
    {
        protected TestPaper() { }

        /// <summary>
        ///
        /// </summary>
        /// <param name="tenantId">所属租户</param>
        /// <param name="displayName"></param>
        public TestPaper(Guid? tenantId, string displayName)
        {
            TenantId = tenantId;
            DisplayName = displayName;
        }

        /// <summary>
        /// 所属租户
        /// </summary>
        public Guid? TenantId { get; set; }

        /// <summary>
        ///
        /// </summary>
        public string DisplayName { get; set; }
    }
```

- 定义一个无参构造函数，例如 protected TestPaper() { }
  - 此构造函数是提供给ORM用来从数据库中获取实体
- 推荐，定义一个主构造函数，例如public TestPaper(Guid? tenantId, string displayName){}
  - 确保实体在创建时的有效性
  - 总是在此构造函数中初始化子集合
  - 不要在此函数中生成guid键，应该将其做成参数获取
- 属性不要加上Virtual关键字
  - virtual关键字主要用于延迟加载，提供性能用的。只有定义成virtual后才可以延迟加载
  - 动态代理
  - 不推荐使用延迟加载，是以往项目中，延迟加载会暴露各种问题。排查起来也麻烦。
- 根据需求继承审计类FullAuditedAggregateRoot<TKey>等
  - 仓库实体定义 public class IdentityUser : FullAuditedAggregateRoot<Guid>{}
- ~~实体都定义对应的仓储接口~~
  - 按照规范来说，只是推荐给聚合根实体定义仓储接口，但是考虑后期来的小伙伴能力及减少复杂度角度考虑
  - 如果我们定义自己的仓储接口，建议方法xxx
- 我们的实体案例
  - 接口

```
public interface ITestPaperRepository :IBasicRepository<TestPaper, Guid>{}
```

  - 实现类
  - http://www.bubuko.com/infodetail-3659620.html

```
 public class TestPaperRepository : EfCoreRepository<JiePeiDbContext, TestPaper, Guid>, ITestPaperRepository
    {
        public TestPaperRepository(IDbContextProvider<JiePeiDbContext> dbContextProvider)
        : base(dbContextProvider)
        {
        }

        public async Task<int> GetAsync()
        {
            await Task.Yield();
```

```
            return 1;
        }
    }
```

二、Dto定义

```
public class CreateBookDto : IValidatableObject
    {
        [Required]
        [StringLength(100)]
        public string Name { get; set; }

        [Required]
        [StringLength(1000)]
        public string Description { get; set; }

        [Range(0, 999.99)]
        public decimal Price { get; set; }

        public IEnumerable<ValidationResult> Validate(
            ValidationContext validationContext)
        {
            if (Name == Description)
            {
                yield return new ValidationResult(
                    "Name and Description can not be the same!",
                    new[] { "Name", "Description" }
                );
            }
        }
    }
```

- 在有**增改**逻辑中，继承IValidatableObject接口，在Dto中判断验证
  - yield return new ValidationResult();
- 多使用数据注解方式来验证Dto

-

  以下是一些内置验证特性：

  - `[CreditCard]`：验证属性是否具有信用卡格式。 需要 JQuery 验证其他方法。
  - `[Compare]`：验证模型中的两个属性是否匹配。
  - `[EmailAddress]`：验证属性是否具有电子邮件格式。
  - `[Phone]`：验证属性是否具有电话号码格式。
  - `[Range]`：验证属性值是否在指定的范围内。
  - `[RegularExpression]`：验证属性值是否与指定的正则表达式匹配。
  - `[Required]`：验证字段是否不为 null。 有关此属性的行为的详细信息，请参阅 [Required] 特性。
  - `[StringLength]`：验证字符串属性值是否不超过指定长度限制。
  - `[Url]`：验证属性是否具有 URL 格式。
  - `[Remote]`：通过在服务器上调用操作方法来验证客户端上的输入。 有关此属性的行为的详细信息，请参阅 [Remote] 特性。

  在 System.ComponentModel.DataAnnotations 命名空间中可找到验证特性的完整列表。

  - 命名空间，不要带Dto，到文件夹级别，比如Jiepei.Intelligent.Molds，而不是Jiepei.Intelligent.Molds.Dto
  - 命名规范
    - Input：CreateOrUpdateStageInput
    - Output：StageListOutput
    - 其他：GetStageListWithGradeDto

三、应用服务接口定义

- vNext 提供了标准基类ApplicationService 和 简单Crud基类 CrudAppService 给我们使用。但是部门内部推荐使用基类ApplicationService ，自己定义扩展。
  - 不推荐基类 CrudAppService的原因是：提供了简单的增删改查方法，重写需要继承，但是如有特殊需求，比如列表查询条件扩展，就很不方便，容易导致方法名称冲突，继而系统异常。
- **接口采用Restful风格**【ABP vNext 动态规则】
  - 如果方法名称以 GetList，GetAll 或 Get 开头，HTTP Method 会被定义为 **Get**
    - GetEntityChanges
  - 如果方法名称以 Put 或 Update 开头，HTTP Method 会被定义为 **Put**
    - UpdateOrganizationUnit
  - 如果方法名称以 Delete 或 Remove 开头，HTTP Method 会被定义为 **Delete**
    - DeleteOrganizationUnit
    - RemoveUserFromOrganizationUnit
  - 如果方法名称以 Create，Add ,Insert 或 Post 开头，HTTP Method 会被定义为**Post**
    - AddUsersToOrganizationUnit
    - CreatePayment

四、枚举定义