**启用多租户**

默认情况多租户是禁用的，我们可以在我们模块的PreInitialize（预初始化）里启用它，如下：

Configuration.MultiTenancy.IsEnabled = true;

**宿主与租户**

首先我们要在多租户系统里定义两个术语：

* Tenant（租户）：系统被分成多个租户，不同租户之间得数据是隔离的
* Host（宿主）：宿主是单例的（就一个宿主），这个宿主负责创建和管理租户，所以“Host user（宿主用户）”拥有更高级别，不依赖于租户，并能控制租户。

**会话（Session）**

* ABP定义了IAbpSession接口，用来获取user（用户）和tenant id(租户Id)。

**数据过滤**

如果实体实现了IMustHaveTenant和ImayHaveTenant两个接口中的一个，ABP上下文在获取数据时，会自动过滤掉非当前租户的数据

**IMustHaveTenant 接口**

该接口通过定义TenantId属性为不同租户区分实体。如下所示，一个实体实现IMustHaveTenant：

public class Product : Entity, **IMustHaveTenant**

{

**public int TenantId { get; set; }**

public string Name { get; set; }

//...other properties

}

因此ABP知道这是一个特定租户的实体并自动与其它租户的实体分离。

**IMayHaveTenant 接口**

我们有时需要在宿主与租户之间共享一个实体，所以一个实体可能是宿主的或租户的。这样我们必须继承IMayHaveTenant接口，如下

public class Role : Entity, **IMayHaveTenant**

{

**public int? TenantId { get; set; }**

public string RoleName { get; set; }

//...other properties

}

靠TenantId属性来区分是宿主实体还是租户实体。如果为null表示这是一个宿主实体，否则它就是一个租户实体，它的值就是租户Id。

**在宿主与租户间切换**

切换到其它租户的数据库，例如：

public class ProductService : ITransientDependency

{

private readonly IUnitOfWorkManager \_unitOfWorkManager;

public ProductService(IRepository<Product> productRepository, **IUnitOfWorkManager unitOfWorkManager**)

{

\_unitOfWorkManager = unitOfWorkManager;

}

[UnitOfWork]

public virtual List<Product> GetProducts(int tenantId)

{

using (**\_unitOfWorkManager.Current.SetTenantId(tenantId)**)

{

}

}

}