

第32期-多租户模块实现

2020年9月24日 13:24

多租户技术或称多重租赁技术，是一种软件架构技术，用于创建 SaaS 应用程序，可在多用户的环境下共用相同的系统或程序组件，并且可确保各用户间数据的隔离性。

多租户简单来说是指一个单独的实例可以为多个组织服务，因为多个租户共享一份系统的核心代码，因此当系统升级时，只需要升级相同的核心代码即可，共享资源可以节约成本。

ABP的多租户模块提供了创建多租户应用程序的基本功能，**理解主机、租户和用户的**关系。

租户模块包

```
Install-Package Volo.Abp.MultiTenancy
```

```
[DependsOn(typeof(AbpMultiTenancyModule))]
```

多租户相关设置

```
Configure<AbpMultiTenancyOptions>(options =>
{
    options.IsEnabled = true;
});
```

数据库架构

共享的单数据库，每个租户一个数据库，混合模式（某些租户共享，某些租户独立）

实体与多租户

```
public class Article : Entity<int>, IMultiTenant
{
    public Guid? TenantId { get; set; }

    public string Title { get; set; }
}

public DbSet<Article> Articles { get; set; }
```

理解主机级别与租户机别

IMultiTenant.TenantId 是可以为null的。如果为null，则表示该实体由主机方拥有，而不由租户拥有。当您在系统中创建供租户和主机双方使用的功能时，此功能很有用。

读取自动，设置手动。通常，您可以使用 ICurrentTenant 来设置。

当前租户

ICurrentTenant 是与多租户基础架构进行交互的主要服务。些其他基类已经已经预先注入 ICurrentTenant 接口：ApplicationService, DomainService, AbpController

属性：Id、Name 和 IsAvailable

```
using (CurrentTenant.Change(tenantId))
{
    return await _productRepository.GetCountAsync();
}
```

CurrentTenant.Change(null) 切换到主机环境。

禁用多租户数据库过滤

```
using (_dataFilter.Disable<IMultiTenant>())
{
    return await _productRepository.GetCountAsync();
}
```

如果租户具有单独的数据库，则此方法将不起作用。

确定当前租户

多租户应用程序的第一件事是确定运行时的当前租户。

默认实现：CurrentUserTenantResolveContributor

Volo.Abp.AspNetCore.MultiTenancy 提供了从 HTTP 请求中获得当前租户的 ID 信息的方法：

QueryStringTenantResolveContributor
FormTenantResolveContributor
RouteTenantResolveContributor

HeaderTenantResolveContributor
CookieTenantResolveContributor

DomainTenantResolveContributor

默认KEY: __tenant

```
services.Configure<AbpAspNetCoreMultiTenancyOptions>(options =>
{
    options.TenantKey = "MyTenantKey";
});
```

使用中间件解析租户

```
app.UseAuthentication()
app.UseMultiTenancy();
```

域名租户解析

```
Configure<AbpTenantResolveOptions>(options =>
{
    options.AddDomainTenantResolver("{0}.xcode.me");
});
```

域名租户测试

C:\Windows\System32\drivers\etc

```
127.0.0.1    tenant1.xcode.me
127.0.0.1    tenant2.xcode.me
```

ipconfig /flushdns

<https://tenant1.xcode.me:5001/api/app/article>

自定义租户解析器

HttpTenantResolveContributorBase->TenantResolveContributorBase->
ITenantResolveContributor

```
using Volo.Abp.MultiTenancy;
```

```
namespace MultiTenancyDemo.Web
{
    public class MyCustomTenantResolveContributor :
    TenantResolveContributorBase
    {
        public override string Name => "Custom";
    }
}
```

```
        public override void Resolve(ITenantResolveContext context)
        {
            //TODO...
        }
    }
}

Configure<AbpTenantResolveOptions>(options =>
{
    options.TenantResolvers.Add(new MyCustomTenantResolveContributor());
});
```