Filter...

latest (v7.1)

ABP Framework

Overall

Introduction
Tutorials & Articles
NLayer Architecture
Module System
Startup Configuration
Multi-Tenancy
OWIN Integration
Debugging
API Reference

Common Structures

Dependency Injection
Session
Caching
Logging
Setting Management
Timing
Object To Object Mapping (and AutoMapper Integration)
Email Sending (and MailKit Integration)

Domain Layer

Entities
Multi-Lingual Entities
Value Objects
Repositories
Domain Services
Specifications
Unit Of Work
Domain Events (EventBus)
Data Filters
Dynamic Parameter System
Object Comparators

Application Layer

Application Services
Data Transfer Objects
Validating Data Transfer Objects

In this document

About Multi-Tenancy
Enabling Multi-Tenancy
Tenant Entity
Tenant Manager
Default Tenant

Edit on GitHub

About Multi-Tenancy

We strongly recommend that you read the multi-tenancy documentation before this one.

Enabling Multi-Tenancy

ASP.NET Boilerplate and Module Zero can run in **multi-tenant** or **single-tenant** modes. Multi-tenancy is disabled by default. We can enable it in the PreInitialize method of our module as shown below:

Copy

```
[DependsOn(typeof(AbpZeroCoreModule))]
public class MyCoreModule : AbpModule
{
    public override void PreInitialize()
    {
        Configuration.MultiTenancy.IsEnabled = true;
    }

    ...
}
```

Note: Even if our application is not multi-tenant, we must define a default tenant (see Default Tenant section of this document).

When we create a project template based on ASP.NET Boilerplate and Module Zero, we have the **Tenant** entity and the **TenantManager** domain service.

Tenant Entity

The Tenant entity represents a Tenant of the application.

Copy

```
public class Tenant : AbpTenant<Tenant, User>
{

}
```

It's derived from a generic **AbpTenant** class. Tenant entities are stored in the **AbpTenants** table in the database. You can add your own custom properties to the Tenant class.

The AbpTenant class defines some base properties, the most important are:

- **TenancyName**: This is the **unique** name of a tenant in the application. It should not normally be changed. It can be used to allocate subdomains to tenants like '**mytenant**.mydomain.com'. As such, it cannot contain spaces. Tenant.**TenancyNameRegex** constant defines the naming rule.
- **Name**: An arbitrary, human-readable, long name of the tenant.
- **IsActive**: True, if this tenant can use the application. If it's false, no user of this tenant can login to the application.

The AbpTenant class inherits **FullAuditedEntity**. This means it has creation, modification and deletion **audit properties**. It is also **Soft-Delete**, so when we delete a tenant, it's not deleted from the database, just marked as deleted.

Finally, the **Id** of AbpTenant is defined as an **int**.

Tenant Manager

The Tenant Manager is a service to perform the **domain logic** for tenants:

Copy

```
public class TenantManager : AbpTenantManager<Tenant, Role, User>
{
    public TenantManager(IRepository<Tenant> tenantRepository)
        : base(tenantRepository)
    {

    }
}
```

The TenantManager is also used to manage the tenant features. You can add your own methods here. You can also override any method of the AbpTenantManager base class for your own needs.

Default Tenant

ASP.NET Boilerplate and Module Zero assume that there is a pre-defined tenant where the TenancyName is '**Default**' and the Id is **1**. In a single-tenant application, this is used as the only tenant. In a multi-tenant application, you can delete it or make it passive.