

# 第26期-审计日志基本配置

2020年9月7日 9:22

## 审计日志

提供在任何特定时间的操作过程或事件产生影响活动顺序的文件证据。

## 构建演示服务

```
public class Apple : Entity<int>
{
    public string Name { get; set; }
}

[ConnectionStringName(ConnectionStringNames.DefaultConnectionStringName)]
public class HelloWorldDbContext : AbpDbContext<HelloWorldDbContext>
{
    public DbSet<Apple> Apples { get; set; }

    public HelloWorldDbContext(DbContextOptions<HelloWorldDbContext>
options) : base(options)
    {
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.ConfigureAppleStore();

        modelBuilder.ConfigurePermissionManagement();

        modelBuilder.ConfigureAuditLogging();
    }
}

public static class HelloWorldDbContextModelCreatingExtensions
{
    public static void ConfigureAppleStore(this ModelBuilder modelBuilder)
    {
        Check.NotNull(modelBuilder, nameof(modelBuilder));

        modelBuilder.Entity<Apple>(b =>
        {
            b.ToTable("Apples", "Foo");
            b.ConfigureByConvention();
        });
    }
}
```

```

        b.HasKey(x=>x.Id);
        b.Property(x => x.Name).IsRequired().HasMaxLength(128);
    });
}

public class AppleDto : EntityDto<int>
{
    public string Name { get; set; }
}

public class AppleAutoMapperProfile : Profile
{
    public AppleAutoMapperProfile()
    {
        CreateMap<Apple, AppleDto>();
        CreateMap<AppleDto, Apple>();
    }
}

Configure<AbpAutoMapperOptions>(options =>
{
    options.AddMaps<FoobarModule>();
});

public class AppleAppService : CrudAppService<Apple, AppleDto, int,
PagedAndSortedResultRequestDto, AppleDto>, IRemoteService
{
    public AppleAppService(IRepository<Apple, int> repository) :
base(repository)
    {
    }
}

public class AppleDataSeederContributor : IDataSeedContributor,
ITransientDependency
{
    private readonly IRepository<Apple, int> _appleRepository;

    public AppleDataSeederContributor(IRepository<Apple, int> appleRepository)
    {
        _appleRepository = appleRepository;
    }

    public async Task SeedAsync(DataSeedContext context)
    {
        if (await _appleRepository.GetCountAsync() <= 0)
        {
            await _appleRepository.InsertAsync(
                new Apple
                {

```

```

                Name = "Apple1"
            },
            autoSave: true
        );

        await _appleRepository.InsertAsync(
            new Apple
            {
                Name = "Apple2"
            },
            autoSave: true
        );
    }
}

```

## 安装模块包

```

abp add-package Volo.Abp.Auditing
Install-Package Volo.Abp.Auditing

```

```
DependsOn(typeof(AbpAuditingModule))]
```

## 管道中插入中间件

```
app.UseAuditing();
```

## 审计日志配置选项

```

Configure<AbpAuditingOptions>(options =>
{
    options.IsEnabled = false; //Disables the auditing system
});

```

**IsEnabled** (默认值: true): 启用或禁用审计系统的总开关。如果值为 false, 则不使用其他选项。

**HideErrors** (默认值: true): 在保存审计日志对象时如果发生任何错误, 审计日志系统会将错误隐藏并写入常规日志。如果保存审计日志对系统非常重要那么将其设置为 false 以便在隐藏错误时抛出异常。

**IsEnabledForAnonymousUsers** (默认值: true): 如果只想为经过身份验证的用户记录审计日志, 请设置为 false。如果为匿名用户保存审计日志, 你将看到这些用户的 UserId 值为 null。

**AlwaysLogOnException**(默认值: true): 如果设置为 true, 将始终在异常/错误情况下保存审计日志, 不检查其他选项(IsEnabled 除外, 它完全禁用了审计日志)。

**IsEnabledForGetRequests** (默认值: false): HTTP GET请求通常不应该在数据库进行任何更改, 审

计日志系统不会为GET请求保存审计日志对象。 将此值设置为 true 可为GET请求启用审计日志系统。

**ApplicationName:** 如果有多个应用程序保存审计日志到单一的数据库, 使用此属性设置为你的应用程序名称区分不同的应用程序日志。

**IgnoredTypes:** 审计日志系统忽略的 Type 列表。 如果它是实体类型, 则不会保存此类型实体的更改。 在序列化操作参数时也使用此列表。

**EntityHistorySelectors:** 选择器列表, 用于确定是否选择了用于保存实体更改的实体类型。 有关详细信息请参阅下面的部分。

**Contributors:** AuditLogContributor 实现的列表。 贡献者是扩展审计日志系统的一种方式。 有关详细信息请参阅下面的"审计日志贡献者"部分。

## 保存实体属性的变更

保存你的所有实体的所有变化将需要大量的数据库空间, 出于这个原因审计日志系统不保存为实体的任何改变, 除非你明确地对其进行配置。

```
Configure<AbpAuditingOptions>(options =>
{
    options.EntityHistorySelectors.AddAllEntities();
});

Configure<AbpAuditingOptions>(options =>
{
    options.EntityHistorySelectors.Add(
        new NamedTypeSelector(
            "MySelectorName",
            type =>
            {
                if (typeof IEntity).IsAssignableFrom(type)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
        )
    );
});
```

例外的设置可以在实体类型上使用: AuditedAttribute 和 DisableAuditingAttribute 标记。

## 启用和禁用服务操作审计日志

- 在Actions 和 应用服务使用：AuditedAttribute 和 DisableAuditingAttribute
- 可以使用 [Audited] 和 IAuditingEnabled 接口

## 启用和禁用实体更改审计日志

以下情况下实体在实体更改审计日志记录中忽略实体：

- 如果将实体类型添加到 AbpAuditingOptions.IgnoredTypes 列表。
- 如果对象不是实体（没有直接或固有的实现 IEntity）
- 如果实体访问级别不是 public 的。

可在实体**类型和属性**上使用：AuditedAttribute 和 DisableAuditingAttribute 标记。

## 审计日志的保存机制

IAuditingStore 是一个接口，用于保存ABP框架的审计日志对象(下面说明)，如果需要将审计日志对象保存到自定义数据存储中，可以在自己的应用程序中实现 IAuditingStore 并在依赖注入系统替换。

如果没有注册审计存储,则使用 SimpleLogAuditingStore，它只是将审计对象写入标准日志系统.

审计日志模块已在启动模板中配置,它将审计日志对象保存到数据库中(支持多个数据库提供程序)，所以大多数时候你并不需要关心 IAuditingStore 是如何实现和使用的。

## 审计日志对象

默认为每个web请求创建一个审计日志对象，审计日志对象可以由以下关系图表示。

