

第12期-配置与选项

2020年7月31日 9:39

配置方式

ABP 框架配置系统与 ASP.NET Core 百分百兼容，没有做任何扩展，基于键值的配置系统。

第11期-使用多环境与配置文件

系统环境变量，关于开发、测试和生产环境的配置，启动类和配置方法的约定。默认配置文件注入与读取，自定义 JSON 和 XML 配置文件，指定环境配置文件，添加内存字典配置，读取配置值与配置节的技巧。

选项模式

第12期-配置选项与配置系统扩展

基于选项的配置，通过委托配置简单选项，自选项配置，在控制器和视图中注入选项，基于名称的选项配置，创建 Entity Framework 扩展配置提供程序，将配置存储到数据库中。

由于零度视频详细解释了选项模式，本文中只会介绍 ABP 增加的一些功能。

```
public override void ConfigureServices(ServiceConfigurationContext context)
{
    context.Services.Configure<AbpAuditingOptions>(options =>
    {
        options.IsEnabled = false;
    });
}
```

可以直接使用 `Configure<...>`,而不是`context.Services.Configure <...>`

自定义选项

```
"MyOptions": {
  "Value1": 88,
  "Value2": true
}
```

```
public class MyOptions
{
    public int Value1 { get; set; }
```

```

        public bool Value2 { get; set; }
    }

    public override void ConfigureServices(ServiceConfigurationContext context)
    {
        Configure<MyOptions>(options => { options.Value1 = 42; options.Value2 =
true; });
        Configure<MyOptions>
(context.Services.GetConfiguration().GetSection("MyOptions"));
    }

    public class MyService : ITransientDependency
    {
        private readonly MyOptions _options;

        public MyService(IOptions<MyOptions> options)
        {
            _options = options.Value;
        }

        public void DoIt()
        {
            var v1 = _options.Value1;
            var v2 = _options.Value2;
        }
    }
}

```

预配置

```

    public override void PreConfigureServices(ServiceConfigurationContext context)
    {
        PreConfigure<MyOptions>(options => { options.Value1 = 88; options.Value2 =
false; });
    }

    public override void ConfigureServices(ServiceConfigurationContext context)
    {
        var options = context.Services.ExecutePreConfiguredActions<MyOptions>();
    }
}

```