**注入会话**

IAbpSession通常被属性注入到需要的类。如果我们使用属性注入，可以使用

NullAbpSession.Instance作为默认值，如上所示：

public class MyClass : ITransientDependency

{

    public IAbpSession AbpSession { get; set; }

    public MyClass()

    {

        AbpSession = NullAbpSession.Instance;

}

    public void MyMethod()

    {

        var currentUserId = AbpSession.UserId;

        //...

}

}

建议在应用层或更上面的层使用IAbpSession（通常我们不在领域层里使用它）。

ApplicationService.AbpController、AbpApiController和其它基类已经完全注入AbpSession。

**会话属性**

AbpSession定义了几个关键属性：

* UserId：当前用户的Id或空（如果没有当前用户），如果调用需要授权的代码，它就不能为空。
* TenantId：当前租户的Id或空（如果没有当前租户：尚未登录或是个宿主用户）。
* ImpersonatorUserId：模拟用户的Id（如果其它用户模拟当前会话），如果不是模拟登录就为空。
* ImpersonatorTenantId：模拟用户的租户的Id（如果其它用户模拟当前会话），如果不是模拟登录就为空。

MultiTenancySide：可能是宿主或租户。

ClaimsAbpSession是IAbpSession接口的**默认实现**。它从当前用户主体的声明获取会话属性，如下

public class ClaimsAbpSession : IAbpSession, ISingletonDependency

{

    public virtual long? UserId

    {

        get

        {

            var userIdClaim = PrincipalAccessor.Principal?.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier);

            if (string.IsNullOrEmpty(userIdClaim?.Value))

            {

                return null;

            }

            long userId;

            if (!long.TryParse(userIdClaim.Value, out userId))

            {

                return null;

            }

            return userId;

        }

    }

    public IPrincipalAccessor PrincipalAccessor { get; set; }

    public ClaimsAbpSession(IMultiTenancyConfig multiTenancy)

    {

        MultiTenancy = multiTenancy;

        PrincipalAccessor = DefaultPrincipalAccessor.Instance;

    }

}

当然，我们也可以自己实现IAbpSession并将其注入到依赖容器中