**定义许可**

为每个操作定义的唯一许可必须得到授权，为使用许可要先定义一个，ABP是按[模块化](http://www.cnblogs.com/kid1412/p/5973274.html" \t "http://www.cnblogs.com/kid1412/p/_blank)设计的，所以不同的模块可以拥有不同的许可，一个模块为了定义它的许可，应当创建一个继承AuthorizationProvider的类。授权供应器示例如下：

public class MyAuthorizationProvider : **AuthorizationProvider**

{

public override void SetPermissions(IPermissionDefinitionContext context)

{

var administration = context.**CreatePermission**("Administration");

var userManagement = administration.**CreateChildPermission**("Administration.UserManagement");

userManagement.**CreateChildPermission**("Administration.UserManagement.CreateUser");

var roleManagement = administration.**CreateChildPermission**("Administration.RoleManagement");

}

}

IPermissionDefinitionContext拥有获取和创建许可的方法。

一个许可包含一些属性：

* Name：一个系统域内的唯一名称，用一个字符串常量，不用可变的字符串，是一个好的做法。在分级里我们更喜欢用.（点）号来命名，但它不是必须的，你设置为你喜欢的名称，唯一的规则是一定要唯一。
* DisplayName：一个本地化的用来之后在UI上显示许可的文本。
* Description：一个本地化的用来之后在UI上显示许可描述的文本。
* MultiTenancySides：在多租户应用里，许可可被租户或宿主使用，这是一个标记性的枚举，所以一个许可可同时被租户和宿主使用。
* DependedFeature：用来表明一个对[Feature](http://www.cnblogs.com/kid1412/p/6006297.html" \t "http://www.cnblogs.com/kid1412/p/_blank)（特色）的依赖，所以这个许可只有在满足Feature（特色）依赖时才会被允许。

一个许可可以有一个父许可和多个子许可，虽然这对于许可检查没有什么作用，但可能有助于在UI上组织许可。

创建一个授权供应器之后，我们应当在我们模块的预初始化方法里注册它：

**Configuration.Authorization.Providers.Add**<MyAuthorizationProvider>();

授权供应器被自动注册到[依赖注入](http://www.cnblogs.com/kid1412/p/5980068.html" \t "http://www.cnblogs.com/kid1412/p/_blank)里，所以一个授权供应器可以注入任何的依赖（如一个仓储），从而可以使用其它的源来定义许可。

**使用 AbpAuthorize特性**

AbpAuthorize（AbpMvcAuthorize用于Mvc控制器，AbpApiAuthorize用于Web Api控制器）特性，是使用许可最简单也是最常用的方式。假设有一个[应用服务](http://www.cnblogs.com/kid1412/p/6003490.html" \t "http://www.cnblogs.com/kid1412/p/_blank)方法如下所示：

**[AbpAuthorize("Administration.UserManagement.CreateUser")]**

public void CreateUser(CreateUserInput input)

{

//A user can not execute this method if he is not granted for "Administration.UserManagement.CreateUser" permission.

}

CreateUser方法不能被没有“Administration.UserManagement.CreateUser”许可的用户调用。

AbpAuthorize特性也检查当前用户是否已登录（使用[IAbp.Session.UsrId](http://www.cnblogs.com/kid1412/p/5986611.html" \t "http://www.cnblogs.com/kid1412/p/_blank)），所以，如果我们为一个方法声明一个AbpAuthorize，它只用来检查用户是否已登录：

**[AbpAuthorize]**

public void SomeMethod(SomeMethodInput input)

{  
　　//用户如果未登录，不能执行这个方法

}

**使用 IPermissionChecker**

尽管AbpAuthorize特性能完美应对大部分情况，但有些情况我们必须要在方法内检查一个许可，这时我们可以注入并使用IPermissionChecker，如下所示：

public void CreateUser(CreateOrUpdateUserInput input)

{

if (!**PermissionChecker.IsGranted("Administration.UserManagement.CreateUser")**)

{

throw new AbpAuthorizationException("You are not authorized to create user!");

}  
　　//一个用户如果没有经过“Administration.usermanagerment.CreateUser”许可的允许，是不可以到达这里的。

}

**客户端（Javascript）**

在客户端里，我们可以使用定义在abp.auth命名空间里的API，大部分情况下，我们需要检查当前用户是否有一个指定的许可（使用许可名称），例如：

abp.auth.isGranted('Administration.UserManagement.CreateUser');