**简介**

每个应用必需存储一些设置，并在应用里的某处使用它。

一个设置是一对name-value字符串，通常存储在数据库（或其它源）。

**关于 ISettingStore**

为使用设置系统,必须实现ISettingStore接口。

**定义设置**

public class MySettingProvider : SettingProvider

{

public override IEnumerable<SettingDefinition> GetSettingDefinitions(SettingDefinitionProviderContext context)

{

return new[]

{

new SettingDefinition(

"SmtpServerAddress",

"127.0.0.1"

),

new SettingDefinition(

"PassiveUsersCanNotLogin",

"true",

scopes: SettingScopes.Application | SettingScopes.Tenant

),

};

}

}

GetSettingDefinitions方法返回SettingDefinition对象集。

SettingDefinition类的构造器有一些参数：

* Name（必需）：一个设置必须有一个系统范围内唯一的名称，最好定义成一个常量，而不是可变的字符串。
* Default value：有一个默认值，可以是null或“”。
* Scope：一个设置应该有它的Scope（见下方）。
* Display name:一个显示在UI上的本地化名称字符串。
* Description：一个显示在UI上的本地化描述字符串
* Group：只用在UI上，不在设置管理里使用，用来群组设置。
* IsVisibleToClients：设为true使客户端上的设置可用。
* isInherited：这个设置是否被租户和用户继承（见“设置 scope”）。
* customData：为这个定义设置一个用户数据。

在创建一个设置供应器后，我们应该为我们的模块，在预初始化方法里注册它。

Configuration.Settings.Providers.Add<MySettingProvider>();

**服务端**

var value1 = await SettingManager.GetSettingValueAsync<bool>("PassiveUsersCanNotLogin");

var value2 = SettingManager.GetSettingValue("SmtpServerAddress");