CrystalWall 权限系统开发者指南 **Version1.0**

权限系统组成

对于任何一个权限系统来说，基本上都分为两个部分。

- 认证：对用户所声称的身份进行合法性验证。
- 授权：对用户声称的身份能够访问的资源进行控制

由于当前开源认证系统非常丰富，例如统一身份认证 OAuther，在 CrystalWall V1.0 版本中不包含认证部分。1.0 版本仅仅针对授权进行开发。

权限

权限是一个非常广泛的概念，什么是权限？权限有非常繁多的种类，例如： admin 用户能访问数据库，论坛匿名用户不能发帖等等。总体上来说，权限就是对系统中某一种资源进行的访问。

对于这种访问，CrystalWall 权限系统使用抽象类 PermissionInfo 类表示。每一种权限都具有一个名称和一个动作字符表示。名称和动作的具体含义有 PermissionInfo 的具体实现类决定。例如 ,你可以设计一个 PagePermissionInfo 类表示对 html 网页的权限 ,他的 name 名称为网页的虚拟路径，action 动作为 browser 浏览等。

对于一个身份来说，一般不会仅仅具有一个或一种权限。例如网站管理员具有上传图片的权限，也具有删帖的权限等。那么如果表示一个身份所具有的所有权限呢？在 CrystalWall 系统中，使用了一个叫 PermissionInfoCollection 的类来表示一个身份具有的“权限集”。一个权限集一般包含同种类型的许多权限。当然，你也可以实现自己的子类以包含多种不同类型的权限。

有了权限和权限集的概念 ,那么如何才能判断一个身份能够访问某种资源呢？这个问题相当简单，一个身份具有一个 PermissionInfoCollection，而对某种资源的访问，CrystalWall 使用 PermissionInfo 表示，那么要判断身份是否能够访问某种资源，只需要判断身份具有的权限集 PermissionInfoCollection 是否包含要访问的资源所代表的 PermissionInfo 即可。

因此，CrystalWall 权限系统在 PermissionInfoCollection 类中设计了一个 Contains 方法用于判断权限集是否包含某个权限。如果包含，表示此身份允许对 PermissionInfo 所代表的资源进行“某种”访问，如果不包含，则表示身份没有得到“授权”不允许访问资源。

PermissionInfo 类

这个类是 CrystalWall 权限系统的核心类。通过以上分析，PermissionInfo 封装了对某种资源的某种访问。实际上，这种“访问”也具有相关的关系！

例如：对磁盘上某个文件进行的“编辑”操作，就隐含表示了对这个文件可以进行“读取查看”操作，而对这个文件的”删除“操作，就不一定包含”读取“操作。为了表示这种在权限之间的”包含关系 “，PermissionInfo 类中也具有一个方法：Contains。但这个方法是抽象的，需要开发者根据具体的权限类自己进行扩充。

注意：在 CrystalWall 权限系统中并没有运用 Composite 模式使得 PermissionInfoCollection 继承自 PermissionInfo，因为我们认为在行为上，权限集与权限已经表现的不尽相同了。记住：一个权限集并不是一个权限，PermissionInfoCollection 与 PermissionInfo 并不是面向对象中所说的“is a"关系。

正向与负向关系

涉及到权限概念，就必然涉及到正向和负向权限的问题。所谓正向和负向权限，类似于我们日常所说的“授予某人能够访问某种资源“和”拒绝某人访问某种资源“的关系。例如，你可以授予管理员访问磁盘文件的权限，但拒绝匿名用户访问磁盘文件的权限。不幸的是，如果在一个权限系统中同时包含正向和负向的权限，系统将会产生严重的冲突。例如：对某人授予某个文件编辑的权限，但是又对某人拒绝对此文件查看的权限，此时由于编辑文件权限又包含查看权限，那么对此用户是能够编辑文件还是不能编辑文件呢？这就是同时具有正向和负向权限必然导致的一个冲突！

对于这种冲突，crystalwall 系统依照惯例优先原则：不建议使用负向权限。即 CrystalWall 系统中的 PermissionInfo 类不处理负向权限，他只表示对某个资源”授予“权限，而没有”拒绝“的权限！由于 PermissionInfo 只是单一方向的授予权限，因此 crystalwall 完全避免了正负向权限的冲突！

对于 PermissionInfoCollection 权限集来说，默认的处理方式是：如果其中有一个权限包含指定权限，则整个权限集包含此权限！默认情况下，PermissionInfoCollection 也始终贯彻了惯例优先原则----使用单向权限！但是，由于实际情况下在人们的概念中的确存在”拒绝“的权限，因此，crystalwall 系统也不随意”拒绝此种实际的思维模式“。crystalwall 在 PermissionInfoCollection 中加入了一个 Elect 委托，这个委托是”选举“的含义，你可以把权限集看成是一个议会，其中的某个权限是一个议员，一个议员包含某种权限表示此议员投了赞成票，而其他议员不包含，则表示此议员投了反对票。此时，crystalwall 使用 Elect 委托来决定权限集是否包含此权限。例如，你可以加入一个少数服从多数委托，当权限集中多数”议员“投反对票时，表示此权限集不包含此权限！

虽然 crystalwall 使用 elect 委托这种折中的办法解决正负权限冲突的问题，但是 crystalwall

强烈建议开发者使用单一的正向权限，这样将会使你的开发工作变的更加方便。

权限点 PermissionPoint

一个系统，在某个希望进行权限控制的运行点就叫做“权限点”。权限点的作用就是告诉调用方：在这个地方要往下运行，必须当前客户具有某个权限，否则不允许继续！在一个系统中，crystalwall 认为权限点都是相对固定的，在系统发布稳定版本之前，系统中需要进行控制的限制点都应该已经固定。因此，crystalwall 所要解决的问题就是如何声明系统中的权限点。

声明权限点的地方，一般就是字段、方法、事件、委托和属性。那么如何在这些元素上定义权限点呢？Crystalwall 采用了元注释标记的方式进行权限点的声明。例如：你可以在某个方法上声明此方法需要进行权限控制：

[PermissionPoint(Type="Crystalwall.FilePermissionPoint" ， Resource="c:\crystall" name="*.txt", action=“delete")]

Public void Delete() {.....}

这个权限点定义表示在执行 delete 方法前，当前用户必须具有对磁盘 c 下 crystall 目录下所有 txt 文件具有 delete 的权限

每一种类型的权限点，都能根据自己的条件生成 PermissionInfo 对象，例如上例中 Crystalwall.FilePermissionPoint 将生成 FilePermissionInfo 权限信息对象。具体生成何种类型，完全由 PermissionPoint 子类决定

【注：crystall 目前支持（开发版）动态权限点定义，即定义权限点时的 name、resource、action 属性可以定义成动态的方法形式，这个方法是权限点被定义的实例类中的方法。例如，在 delete 方法中，资源的路径是动态获取的，你可以这样定义：

```
Public class MyAction {
[DynamicPermissionPoint(Type="Crystalwall.FilePermissionPoint"                    ,
Resource="GetResource()" name="GetName(0, 2)", action= “delete")]
Public void delete(object p1, object p2, object p3, object p4) {.....}
//获取资源的方法;
Public string GetResource() {....};
}
```

动态的权限点定义，将使用 MyAction 的 GetResource 方法获取资源 id，使用 GetName 方法并传递 delete 的 p1, p3 参数获取名称来生成 PermissionInfo 从而进行动态的权限判定

注意：这种动态的权限点定义尚在开发中】

权限点的织入

权限点已经在代码中定义完成，那么 crystall 是如何知晓这些权限点的呢？Crystalwall 是通过两种方式：

1．静态织入。直接通过编译器修改 IL 代码（需要 aspectdng 的支持）

系统目前采用 aspectdng 做静态植入(lib 目录下 aspectdng.dll ,在编译项目时 ,visual studio 中为项目属性的 “生成事件” 选项卡---》后期生成事件行中添加 “aspectdng.dll CrystalWall.dll）即可

2．动态代理：使用动态代理技术从代理中获取受控对象进行权限控制（尚在开发中）

手动进行权限控制

Crystall 通过静态定义权限点的方式可以进行方法级的权限控制。但是，又是用户需要在方法内部，代码运行的某一行开始进行权限控制该怎么办呢？此时， crystall 提供了 PermissionInfo 类重载运算符"!!"进行手动权限控制：

```
Public void YourMethod() {
  .....
  FilePermissionInfo f = new FilePermission("");
  f!!; //如果当前用户不具有 f 权限，则抛出权限异常
```