**简介**

通知用来**告知**用户系统里特定的事件发生了，ABP提供一个**发布/订阅**，它基于实时通知基础框架。

**发送模式**

有两种方式可以发送通知给用户：

* 用户**订阅**一个特定的通知类型，然后我们**发布**一个此类型的通知，它会分发给所有订阅的用户，这就是**发布/订阅**模式。
* 我们可以**直接发送**一个通知给目标用户（users）。

**订阅通知**

INotificationSubscriptionManager提供了用来订阅通知的API，例如：

public class MyService : ITransientDependency

{

private readonly INotificationSubscriptionManager \_notificationSubscriptionManager;

public MyService(**INotificationSubscriptionManager notificationSubscriptionManager**)

{

\_notificationSubscriptionManager = notificationSubscriptionManager;

}

//订阅一个一般通知

public async Task Subscribe\_SentFrendshipRequest(int? tenantId, long userId)

{

**await \_notificationSubscriptionManager.SubscribeAsync(new UserIdentifier(tenantId, userId), "SentFrendshipRequest");**

}

//订阅一个实体通知

public async Task Subscribe\_CommentPhoto(int? tenantId, long userId, Guid photoId)

{

**await \_notificationSubscriptionManager.SubscribeAsync(new UserIdentifier(tenantId, userId), "CommentPhoto", new EntityIdentifier(typeof(Photo), photoId));**

}

}

首先，我们注入**INotificationSubscriptionManager**，第一方法订阅一个一般通知，当某人发送一个好友请求时，用户会收到通知，第二个订阅一个与**特定实体**（Photo)相关的通知，当某人给特定照片写评论后，用户会收到通知。

每一个通知类型都应当有一个唯一的名称（如示例中的SentFrendshipRequest 和CommentPhoto）。

**发布通知**

**INotificationPublisher**用来发布通知，例如：

public class MyService : ITransientDependency

{

private readonly INotificationPublisher \_notiticationPublisher;

public MyService(**INotificationPublisher notiticationPublisher**)

{

\_notiticationPublisher = notiticationPublisher;

}

//发送一个一般通知给一个特定用户

public async Task Publish\_SentFrendshipRequest(string senderUserName, string friendshipMessage, UserIdentifier targetUserId)

{

**await \_notiticationPublisher.PublishAsync("SentFrendshipRequest", new SentFrendshipRequestNotificationData(senderUserName, friendshipMessage), userIds: new[] { targetUserId });**

}

//发送一个实体通知给一个特定用户

public async Task Publish\_CommentPhoto(string commenterUserName, string comment, Guid photoId, UserIdentifier photoOwnerUserId)

{

**await \_notiticationPublisher.PublishAsync("CommentPhoto", new CommentPhotoNotificationData(commenterUserName, comment), new EntityIdentifier(typeof(Photo), photoId), userIds: new[] { photoOwnerUserId });**

}

//发送一个一般通知给所有当前租户（在会话里）里的订阅它的用户

public async Task Publish\_LowDisk(int remainingDiskInMb)

{

//Example "LowDiskWarningMessage" content for English -> "Attention! Only {remainingDiskInMb} MBs left on the disk!"

**var data = new LocalizableMessageNotificationData(new LocalizableString("LowDiskWarningMessage", "MyLocalizationSourceName"));**

**data["remainingDiskInMb"] = remainingDiskInMb;**

**await \_notiticationPublisher.PublishAsync("System.LowDisk", data, severity: NotificationSeverity.Warn);**

}

}

在第一个例子里， 我们发布一个通知给一个单独的用户，SentFrendshipRequestNotificationData应该继承于**NotificationData**，如下所示：

[Serializable]public class SentFrendshipRequestNotificationData : **NotificationData**

{

public string SenderUserName { get; set; }

public string FriendshipMessage { get; set; }

public SentFrendshipRequestNotificationData(string senderUserName, string friendshipMessage)

{

SenderUserName = senderUserName;

FriendshipMessage = friendshipMessage;

}

}

**客户端**

当接收到一个实时的通知，ABP在客户端触发一个全局事件，你可以用如下的方式注册来得到通知：

abp.event.on('**abp.notifications.received**', function (userNotification) {

console.log(userNotification);

});

每接收到一个实时通知，都会触发abp.notifications.received事件事件，你可以像上面那样注册该事件来获取通知，查看[javascript事件总线](http://www.cnblogs.com/kid1412/p/6019011.html" \l "Event" \t "http://www.cnblogs.com/kid1412/p/_blank)文档获取事件更多信息。一个收到的“System.LowDisk”通知的Json示例：

{

"userId": 2,

"state": 0,

"notification": {

"notificationName": "System.LowDisk",

"data": {

"message": {

"sourceName": "MyLocalizationSourceName",

"name": "LowDiskWarningMessage"

},

"type": "Abp.Notifications.LocalizableMessageNotificationData",

"properties": {

"remainingDiskInMb": "42"

}

},

"entityType": null,

"entityTypeName": null,

"entityId": null,

"severity": 0,

"creationTime": "2016-02-09T17:03:32.13",

"id": "0263d581-3d8a-476b-8e16-4f6a6f10a632"

},

"id": "4a546baf-bf17-4924-b993-32e420a8d468"

}