**🛡️ Authorization Policy: IsHostRequirement in ASP.NET Core**

**Overview**

This custom authorization policy ensures that a user is allowed to perform certain actions only if they are the **host of an activity**.

**🔧 Components**

**1. IsHostRequirement Class**

public class IsHostRequirement : IAuthorizationRequirement

{

}

* Marker class used to define a custom requirement.
* Implements IAuthorizationRequirement without any additional properties—purely a flag for the policy.

**2. IsHostRequirementHandler Class**

public class IsHostRequirementHandler(AppDBContext dBContext, IHttpContextAccessor httpContextAccessor)

: AuthorizationHandler<IsHostRequirement>

{

protected override async Task HandleRequirementAsync(

AuthorizationHandlerContext context,

IsHostRequirement requirement)

{

var userid = context.User.FindFirstValue(ClaimTypes.NameIdentifier);

if (userid == null) return;

var httpCntxt = httpContextAccessor.HttpContext;

if (httpCntxt?.GetRouteValue("id") is not string activityId) return;

var Attendee = dBContext.ActivityAttendee

.AsNoTracking()

.SingleOrDefault(r => r.UserId == userid && r.ActivityId == activityId);

if (Attendee == null || !Attendee.IsHost) return;

context.Succeed(requirement);

}

}

**🔍 What’s Happening Here?**

* **Extract User ID**: Gets the authenticated user's ID from claims.
* **Extract Activity ID**: Gets the id route parameter from the current HTTP request.
* **Query DB**: Checks the ActivityAttendee table to verify if this user is associated with the activity and is marked as the host.
* **Authorize**: If all checks pass and the user is the host, context.Succeed(requirement) grants authorization.

**🔗 Usage in Startup.cs or Program.cs**

Register the handler and define the policy:

services.AddAuthorization(options =>

{

options.AddPolicy("IsActivityHost", policy =>

policy.Requirements.Add(new IsHostRequirement()));

});

services.AddScoped<IAuthorizationHandler, IsHostRequirementHandler>();

**🧪 Applying the Policy**

Use the [Authorize(Policy = "IsActivityHost")] attribute in your controller:

[Authorize(Policy = "IsActivityHost")]

[HttpPut("{id}")]

public async Task<IActionResult> EditActivity(string id, EditActivityDto dto)

{

// Only the host can access this endpoint

}

**📝 Final Notes**

* Make sure the id route param exists in your route (e.g., /api/activities/{id}).
* Ensure IHttpContextAccessor is added via:
* services.AddHttpContextAccessor();

Let me know if you want this saved as a downloadable PDF or formatted for Markdown, Notion, Confluence, etc.