**LAB-ENUMNS WITH Grpc Protobuff**

In .NET 6, using gRPC with enums is quite straightforward. Below is an example that demonstrates how to use enums in gRPC services with .NET Core 6.

**Step 1: Create a .proto file defining the gRPC service and enum**

syntax = "proto3";

option csharp\_namespace = "GrpcEnumExample";

package example;

// Define an enum

enum Status {

UNKNOWN = 0;

PENDING = 1;

COMPLETED = 2;

FAILED = 3;

}

// Define a message that uses the enum

message Task {

int32 id = 1;

string name = 2;

Status status = 3; // Enum field

}

// Define the gRPC service

service TaskService {

rpc GetTask (TaskRequest) returns (Task);

}

message TaskRequest {

int32 task\_id = 1;

}

This proto file defines:

* An enum Status with different values (UNKNOWN, PENDING, COMPLETED, FAILED).
* A Task message that includes an enum field status.
* A TaskService service that provides a GetTask RPC.

**Step 2: Create the gRPC Service in .NET Core 6**

1. **Create a .NET 6 gRPC project:**

In your terminal, run:

dotnet new grpc -n GrpcEnumExample

cd GrpcEnumExample

1. **Add the .proto file:**

Add the .proto file to the Protos directory. You can place the file in Protos/task.proto.

1. **Modify the Program.cs to register gRPC services:**

Here’s a basic example of Program.cs:

**using GrpcEnumExample;**

using Microsoft.AspNetCore.Builder;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);

// Add gRPC services to the container.

**builder.Services.AddGrpc();**

var app = builder.Build();

// Configure the HTTP request pipeline.

**app.MapGrpcService<TaskService>();**

app.Run();

1. **Implement the gRPC service in TaskService.cs:**

In Services/TaskService.cs, implement the TaskService service by inheriting from the generated TaskServiceBase:

using Grpc.Core;

using GrpcEnumExample;

public class TaskService : TaskService.TaskServiceBase

{

// Simulate a task lookup based on task\_id

public override Task<Task> GetTask(TaskRequest request, ServerCallContext context)

{

var task = new Task

{

Id = request.TaskId,

Name = "Sample Task",

Status = Status.PENDING // Enum usage here

};

return Task.FromResult(task);

}

}

**Step 3: Create a gRPC Client to Call the Service**

1. **Add gRPC client dependencies:**

Add the NuGet packages to your client project:

dotnet add package Grpc.Net.Client

dotnet add package Google.Protobuf

dotnet add package Grpc.Tools

1. **Create the client to call the gRPC service:**

Here's how you would call the service in a client:

using Grpc.Net.Client;

using GrpcEnumExample;

var channel = GrpcChannel.ForAddress("https://localhost:5001");

var client = new TaskService.TaskServiceClient(channel);

var taskRequest = new TaskRequest { TaskId = 1 };

var task = await client.GetTaskAsync(taskRequest);

Console.WriteLine($"Task ID: {task.Id}");

Console.WriteLine($"Task Name: {task.Name}");

Console.WriteLine($"Task Status: {task.Status}"); // This will print the enum value

**Step 4: Run the Application**

1. Start the server using:
2. dotnet run
3. Create a separate client application, then run it to call the GetTask method.

When you run the client, it will print the task details, including the status, which is an enum value (PENDING in this case).

**Expected Output**

On the client side, when the task is fetched, it will print:

Task ID: 1

Task Name: Sample Task

Task Status: PENDING

**Summary**

This is a simple example of how to use enums with gRPC in .NET Core 6. The core idea is that you define your enums in a .proto file, generate the corresponding classes, and use those enums in your service implementations and client applications.