HABIB UL REHMAN

01

AZURE

DEPENDENCY INJECTION IN AZURE FUNCTION

@HABIBDEVELOPER

FINCHSHIP.COM

Prerequisites

Before you can use dependency injection, you must install the following NuGet packages:

- Microsoft.Azure.Functions.Extensions
- Microsoft.NET.Sdk.Functions package version 1.0.28 or later
- Microsoft.Extensions.DependencyInjection (currently, only version 2.x or later supported)

HABIB UL REHMAN

03

Registering Services

In Azure Functions, you can register services using the Startup class. This class allows you to configure services for dependency injection during function startup.

```
// Example of DI registration in Azure Functions
[assembly: FunctionsStartup(typeof(MyNamespace.Startup))]
namespace MyNamespace
{
    public class Startup : FunctionsStartup
    {
        public override void Configure(IFunctionsHostBuilder builder)
        {
            builder.Services.AddSingleton<IMyService, MyService>();
        }
    }
}

```

HABIB UL REHMAN

04

Injecting Services into Functions

Once services are registered, you can inject them into your Azure Functions using constructor injection or method injection.

```
// Example of injecting services into Azure Functions
public class MyFunction
{
    private readonly IMyService _myService;
    public MyFunction(IMyService myService)
    {
        _myService = myService;
    }
    [FunctionName("MyFunction")]
    public async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
        ILogger log)
    {
        // Use injected service
        var result = await _myService.DoSomethingAsync();
        return new OkObjectResult(result);
    }
}
```

HABIB UL REHMAN

Service lifetimes

Azure Functions apps provide the same service lifetimes as ASP.NET Dependency Injection.

```
// Example of DI registration in Azure Functions
[assembly: FunctionsStartup(typeof(MyNamespace.Startup))]
namespace MyNamespace
{
    public class Startup : FunctionsStartup
    {
        public override void Configure(IFunctionsHostBuilder builder)
        {
            //Transient
            builder.Services.AddTransient<IMyService, MyService>();
            //Scoped
            builder.Services.AddScoped<IMyService, MyService>();
            //Singleton
            builder.Services.AddSingleton<IMyService, MyService>();
        }
    }
}
```

Benefits of Dependency Injection

- Promotes modularity and separation of concerns
- Facilitates unit testing by enabling easy mocking of dependencies
- Enhances code reusability and maintainability
- Supports inversion of control (IoC) principles for decoupled architecture

HABIB UL REHMAN

07

USEFUL?

REPOST AND SHARE WITH FRIENDS TO ACCELERATE THEIR SKILLS

follow for more updates

@HABIBDEVELOPER

FOLLOW

FINCHSHIP.COM