

Dependency Injection in ASP.NET Core | AddSingleton, AddScoped and Add Transient

.

What is Dependency Injection?

- ▶ The Dependency Injection is a process of injecting the object of a class into a class that depends on it.
- ▶ The Dependency Injection is the most commonly used design pattern nowadays to remove the dependencies between the objects. So, the Dependency Injection design pattern allows us to develop loosely coupled software components.

Implement DI:-

The screenshot displays the Visual Studio IDE with the following components:

- File Explorer:** Shows the project structure with files like `EmployeeController.cs`, `EmployeeRepository.cs`, `IEmployeeRepository.cs`, `Employee.cs`, `Startup.cs`, `Object Browser`, and `HomeController.cs`.
- Code Editor:** Displays the `EmployeeController.cs` file. The code is as follows:

```
9 {
10     public class EmployeeController : Controller
11     {
12         //Without DI
13         //public JsonResult GerEmployeeId(int id)
14         //{
15             //    IEmployeeRepository ob = new EmployeeRepository(); //This is Tightly couple with EmployeeRepositor
16             //    Employee emp= ob.GetEmployee(id);
17             //    return Json(emp);
18         //}
19         //With DI this is called Constructor Dependency Injection
20         private readonly IEmployeeRepository _repo = null;
21         public EmployeeController(IEmployeeRepository _repo)
22         {
23             this._repo = _repo;
24         }
25         public JsonResult GerEmployeeId(int id)
26         {
27
28             Employee emp = _repo.GetEmployee(id);
29             return Json(emp);
30         }
31     }
32 }
33 }
```
- Output Window:** Shows the build output. The first build started successfully. The second build failed with the error: `1>----- Build started: Project: AspDotnetCore_Project1, Configuration: Debug Any CPU ----- 1>AspDotnetCore_Project1 -> C:\Users\Lenovo\Desktop\Asp.net_Core\Examples\AspDotnetCore_Project1\bin\Debug\netcoreapp2.1\AspDotnetCore_Project1.dll ===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====`
- Taskbar:** Shows the Windows taskbar with various application icons and the system clock displaying 10:47 on 18-01-2021.

An unhandled exception occurred while processing the request.

InvalidOperationException: Unable to resolve service for type 'AspDotnetCore_Project1.Models.IEmployeeRepository' while attempting to activate 'AspDotnetCore_Project1.Controllers.EmployeeController'.

Microsoft.Extensions.DependencyInjection.ActivatorUtilities.GetService(IServiceProvider sp, Type type, Type requiredBy, bool isDefaultParameterRequired)

Stack Query Cookies Headers

InvalidOperationException: Unable to resolve service for type 'AspDotnetCore_Project1.Models.IEmployeeRepository' while attempting to activate

How to register a Service with ASP.NET Core Dependency Injection Container?

- ▶ We need to register a service with **ASP.NET Core Dependency Injection** Container within the **ConfigureServices()** method of the **Startup** class.
- ▶ Before we can discuss how to register a service with the Dependency Injection Container, it is important to understand the lifetime of service.
- ▶ When a class receives the dependency object through dependency injection, then whether the instance it receives is unique to that instance of the class or not depends on the lifetime of the service. Setting the lifetime of the dependency object determines how many times the dependency object needs to be created.

What are the different methods ASP.NET Core Provides to register a service with Dependency Injection Container?

- ▶ The ASP.NET core provides 3 methods to register a service with the ASP.NET Core Dependency Injection container. The method that we use to register a service will determine the lifetime of that service.
 - ▶ AddSingleton
 - ▶ AddScoped
 - ▶ AddTransient

AddSingleton()

- ▶ When we use the AddSingleton() method to register a service, then it will create a singleton service.
- ▶ It means a single instance of that service is created and that singleton instance is shared among all the components of the application that require it.
- ▶ That singleton service is created when we requested for the first time.



AddScoped()

- ▶ Scoped means instance per request.
- ▶ When we use the AddScoped() method to register a service, then it will create a Scoped service.
- ▶ It means, an instance of the service is created once per each HTTP request and uses that instance in other calls of the same request.



AddTransient()

- ▶ When we use the AddTransient() method to register a service, then it will create a Transient service.
- ▶ It means a new instance of the specified service is created each time when it is requested and they are never shared.

Visual Studio interface showing the development of an ASP.NET Core application. The main window displays the code for `EmployeeController.cs`, which implements a controller using constructor dependency injection.

```
9 {
10     public class EmployeeController : Controller
11     {
12         //Without DI
13         //public JsonResult GerEmployeeId(int id)
14         //{
15         //    IEmployeeRepository ob = new EmployeeRepository(); //This is Tightly couple with EmployeeRepositor
16         //    Employee emp= ob.GetEmployee(id);
17         //    return Json(emp);
18         //}
19         //With DI this is called Constructor Dependency Injection
20         private readonly IEmployeeRepository _repo = null;
21         public EmployeeController(IEmployeeRepository _repo)
22         {
23             this._repo = _repo;
24         }
25         public JsonResult GerEmployeeId(int id)
26         {
27
28             Employee emp = _repo.GetEmployee(id);
29             return Json(emp);
30         }
31     }
32 }
33
```

The right sidebar shows the Output window with the following build log:

```
Build started...
1>----- Build
started:
Project:
AspDotnetCore_P
roject1,
Configuration:
Debug Any CPU
-----
1>AspDotnetCore_P
roject1 -> C:
\Users\Lenovo
\Desktop
\Asp.net_Core
\Examples
\AspDotnetCore_
Project1\bin
\Debug
\netcoreapp2.1
\AspDotnetCore_
Project1.dll
===== Build:
1 succeeded, 0
failed, 0 up-
to-date, 0
skipped
=====
```

The bottom status bar indicates "Build succeeded". The Windows taskbar at the bottom shows the system clock as 10:56 on 18-01-2021.

- Add Singleton service inside the Startup.cs

The screenshot displays the Visual Studio IDE with the following components:

- Menu Bar:** File, Edit, View, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Toolbar:** Includes icons for file operations, debugging, and search.
- Server Explorer:** Shows the project structure for 'AspDotnetCore_Project1'.
- Code Editor:** Displays the 'Startup.cs' file with the following code:

```
16 // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?Li
17 public void ConfigureServices(IServiceCollection services)
18 {
19     //create mvc project add this services first
20     // services.AddMvc();
21     //Create Core MVC project add this service
22     //services.AddMvcCore();//Error
23     var builder = services.AddMvcCore();
24     builder.AddJsonFormatters();
25     services.AddSingleton<IEmployeeRepository, EmployeeRepository>();
26
27 }
28
29 // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
30 public void Configure(IApplicationBuilder app, IHostingEnvironment env)
31 {
32     if (env.IsDevelopment())
33     {
34         app.UseDeveloperExceptionPage();
35     }
36     //Add Default root to configure the Index page from Home Controller
37     app.UseMvcWithDefaultRoute();
38     //app.Run(async (context) =>
39     //{
40         await context.Response.WriteAsync("Hello World!");
41     //}
```
- Object Browser:** Shows the 'ConfigureServices(IServiceCollection services)' method.
- Output Window:** Displays the build output:

```
Build started...
1>----- Build
started:
Project:
AspDotnetCore_P
roject1,
Configuration:
Debug Any CPU
-----
1>AspDotnetCore_P
roject1 -> C:
\Users\Lenovo
\Desktop
\Asp.net_Core
\Examples
\AspDotnetCore_
Project1\bin
\Debug
\netcoreapp2.1
\AspDotnetCore_
Project1.dll
===== Build:
1 succeeded, 0
failed, 0 up-
to-date, 0
skipped
=====
```
- Status Bar:** Shows 'Build succeeded' and 'No issues found'.
- Taskbar:** Includes the Windows Start button, search bar, and various application icons.

Thanks