03

Injetando uma classe como dependência de outra

Você está desenvolvendo uma aplicação da área de Recursos Humanos usando ASP.NET Core 2.0 MVC, e acaba de criar uma nova classe chamada `Calendario`. Essa classe possui métodos que serão úteis para obter dados para várias páginas. Os métodos são:

- `public int GetDiasCorridos(DateTime dataInicial, DateTime dataFinal)`
- `public int GetDiasUteis(DateTime dataInicial, DateTime dataFinal)`
- `public DateTime GetProximasFerias(int funcionarioId)`

Você deseja que uma instância da classe `Calendario` esteja disponível para uso em todos os pontos da sua aplicação, e deseja que o código não referencie a classe diretamente, mas sim através da sua interface `ICalendario`. Quais os passos necessários para realizar essa tarefa?

Selecione 4 alternativas

A Criando um parâmetro `Calendario` no construtor de cada classe que utilizará a instância da classe `Calendario`:

```
public class MinhaClasse()
{
    public MinhaClasse(Calendario calendario)
    {
    }
}
```

B Registrando a classe no container de injeção de dependência, no método `Startup.ConfigureServices`:

```
services.AddTransient<Calendario>();
```

C Criando um parâmetro `ICalendario` no construtor de cada classe que utilizará a instância da classe `Calendario`:

```
public class MinhaClasse()
{
    public MinhaClasse(ICalendario calendario)
    {
    }
}
```

D Usando os métodos da instância do campo `calendario` nos métodos da classe `MinhaClasse`:

```
public void AlgumMetodo(int funcionarioId, DateTime dataInicial, DateTime dataFinal)
{
    DateTime inicioProximasFerias = calendario.GetProximasFerias(funcionarioId)
    int diasCorridos = calendario.GetDiasCorridos(dataInicial, dataFinal)
    int diasUteis = calendario.GetDiasUteis(dataInicial, dataFinal)
}
```

E

Criando uma instância de `Calendario` e acessando os métodos dessa instância nos métodos da classe `MinhaClasse`:

```
public void AlgumMetodo(int funcionarioId, DateTime dataInicial, DateTime dataFinal)
{
    ICalendario calendario = new Calendario();
    DateTime inicioProximasFerias = calendario.GetProximasFerias(funcionarioId)
    int diasCorridos = calendario.GetDiasCorridos(dataInicial, dataFinal)
    int diasUteis = calendario.GetDiasUteis(dataInicial, dataFinal)
}
```

F

Atribuindo o valor do parâmetro `ICalendario` a um campo privado da classe `MinhaClasse`:

```
public class MinhaClasse()
{
    private readonly ICalendario calendario;

    public MinhaClasse(ICalendario calendario)
    {
        this.calendario = calendario;
    }
}
```

G

Registrando a interface no container de injeção de dependência, no método `Startup.ConfigureServices`:

```
services.AddTransient<ICalendario>();
```

H

Registrando a interface e a classe no container de injeção de dependência, no método `Startup.ConfigureServices`:

```
services.AddTransient<ICalendario, Calendario>();
```