表5.1　よく利用されるDIサービス

サービス	内容	既定で利用できるか？	備考
IHostingEnvironment、IHostEnvironment	ホストプロセスに関する情報を提供	○	Blazor WASMでは IWebAssemblyHostEnvironmentを利用する
IConfiguration	構成設定に関する情報を提供	○	サーバーシャットダウンまで同一オブジェクトを使いまわし続ける
ILogger、ILoggerFactory	ロギングを行なうサービス	○	`ILogger<T>`で受け取って利用する
HttpClient、IHttp**Client**Factory	HTTP呼び出しを行なうサービス	×（要自力登録）	IHttp**Client**Factoryを利用したほうがよい
NavigationManager	画面遷移を行なうサービス	○	
DbContext、DbContextFactory	DBアクセスを行なうサービス	×（要自力登録）	DbContextFactoryを利用したほうがよい
IJSRuntime	JavaScriptを呼び出すサービス	○	サーバー側からブラウザ上のJavaScript ランタイムを呼び出す場合に利用できる

5.2.3　自作のサービスをDIコンテナに登録する方法

また、自作のサービスをDIサービスとしてDIコンテナに登録することも可能です。本書では最終章の第11章でBlazor Server、WASMの両方に対応するアプリを開発する際にしか利用していませんが、DIコンテナをより深く理解する一助となるため、ここで簡単に説明しておきます。

一般的に、DIコンテナにサービスを登録したい場合、サービスの有効期間、つまり同じインスタンスをどのような期間だけ使いまわすのか（どのタイミングで別のインスタンスを作るのか）を決める必要があります。サービスの有効期間には**表5.2**の3種類があり、サービスの機能的な要件に応じてどれを使用するのかを選択します。

表5.2　インスタンス生存期間によるDIサービス登録方法の違い

有効期間	一時的（Transient）	スコープ（Scoped）	シングルトン（Singleton）
概要	ユーザーがDIコンテナ（サービスコンテナ）へ要求するたび、新規にインスタンスが生成されるようにする	ユーザーからのリクエスト（接続）のたび、新しいインスタンスが作成されるようにする	同じサーバー内ではシャットダウンまで同じインスタンスを使い続けるようにする
Blazor Serverの場合	ページが切り替わるつど新しいインスタンスがDIコンテナにより生成され、それを利用する	同一セッション中は同一オブジェクトを使い続ける	サーバーシャットダウンまで同一オブジェクトを使いまわし続ける
DIコンテナへの登録方法	`builder.Services.AddTransient<IServiceClass, ServiceClass>();`	`builder.Services.AddScoped<IServiceClass, ServiceClass >();`	`builder.Services.AddSingleton<IServiceClass, ServiceClass >();`

この3つの使い分けは非常に重要で、誤った方法でサービスを登録すると、同一のDIサービスインスタンスを別のページや処理で再利用してしまうことになり、アプリが正しく動作しなくなります。自作のサービスをDIコンテナに登録する場合には、自力でこれらの中から正しい選択をしなければならないのは当然ですが、ASP.NET Coreランタイムが提供している一般的なDIサービス（たとえば`DbContextFactory`など）も上記の方法で登録させるのは、誤った方法でのサービス登録を誘発しや