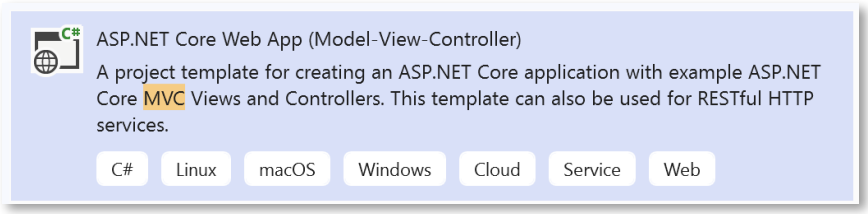
Modul 03: ASP.NET Core MVC App erstellen

Ziel:	*Ziel dieser Übung ist es ein Frontend mit MVC zu erstellen.*
Tools:	*Visual Studio 2022*
Dauer:	*~30min*

1 Neues ASP.Net Core MVC-Projekt anlegen

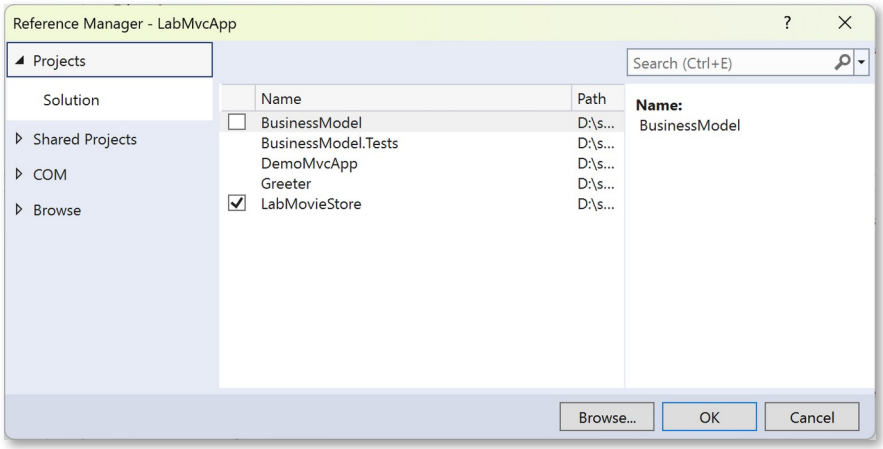
Legen Sie in Ihrer Solution ein neues MVC-Projekt **MovieMvcApp** an.



(Die ASP.Net Webkomponenten müssen installiert sein, damit das Projekttemplate verfügbar wird)

2 Abhängigkeiten einrichten

Legen Sie eine neue Klassenbibliothek **LabMovieStore** an, welche den vorgegebenen Code enthält. Fügen Sie dem MVC-Projekt **MovieMvcApp** dann diese Assembly als Projektreferenz hinzu.

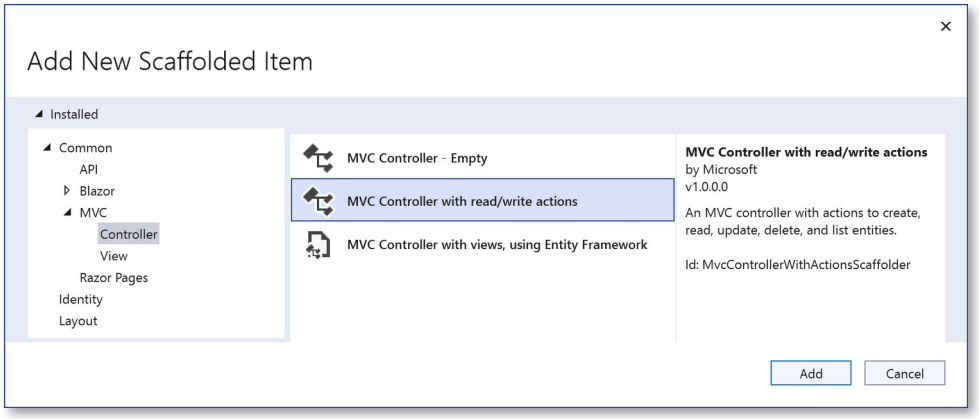


Registrieren Sie in der **Program.cs** den MovieService (vorerst) als Singleton, damit die Filme im Speicher gehalten werden.

```
builder.Services.AddSingleton<IMovieService, MovieService>();
```

3 Controller anlegen

Legen Sie einen neuen Controller namens **MoviesController.cs** an. Sie können als Vorlage read/write actions wählen, um die Methoden-Rümpfe nicht manuell anlegen zu müssen.



Erstellen Sie einen Konstruktur (Code-Snippet `ctor`) und injizieren Sie den `IMovieService`.

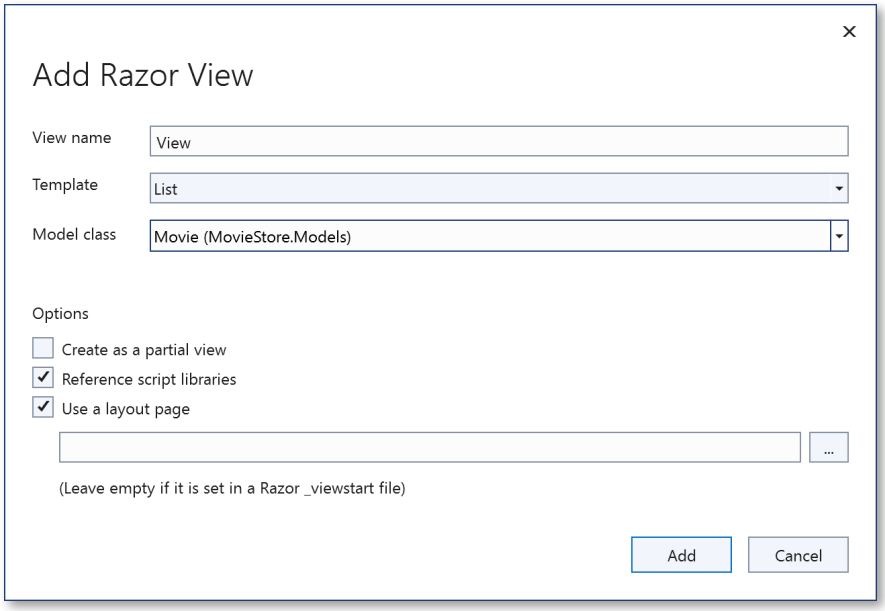
```
private readonly IMovieService _movieService;

public MoviesController(IMovieService movieService)
{
    _movieService = movieService;
}


// Action-Methoden folgen...
```

Implementieren Sie die Action-Methoden entsprechend, welche die Aufrufe auf den IMovieService delegieren. Beachten Sie die Unterscheidung zwischen **GET-** und **POST-Requests**. Die Get-Methoden greifen immer nur lesend auf die Daten zu. Beim Erstellen neuer Daten geben Sie bestenfalls ein leeres Formular zurück. Nur Post-Requests (oder PUT/DELETE) dürfen Daten verändern. Die Antwort der Aktion ist in der Regel eine Weiterleitung auf eine lesende Methode, beispielsweise die Index-Seite (Listenansicht).

4 Ansichten erstellen

Die Darstellung der Datenmodelle findet in Ansichten (sog. Views) statt. Erstellen Sie im Order **Views** den Unterordner **Movies**. Achten Sie darauf, dass der Name genauso lauten wie **Movies**Controller, da ASP.NET Core die Ansichten nicht finden kann. Legen Sie dann jeweils eine View für die Listenansicht (**Index.cshtml**) und für die Details (**Details.cshtml**) an.

Mi dem Scaffold Feature wird der HTML-Code für die Views automatisch generiert und kann nach belieben angepasst werden. Wählen Sie hierfür im Dialog *MVC > View > Razor View*. In der nächsten Ansicht wählen Sie die Art des Templates (jeweils Liste und Details) und die Model-Klasse dazu.



Damit die Navigation zur Detail-Ansicht funktioniert, müssen Sie die Html-Links anpassen. Diese könnten dann so aussehen:

```
<a role="button" class="btn btn-outline-dark" asp-action="Details"
   asp-route-id="@item.Id">Details</a>
<a role="button" class="btn btn-outline-danger" asp-action="Delete"
   asp-route-id="@item.Id">Delete</a>
```

Damit sollten Listenansicht und Details jetzt funktionieren.