Modul 05: ASP.NET Core Forms und Validierung

Ziel:	*Ziel dieser Übung ist es ein Formular zum Hinzufügen von Filmen zu erzeugen inkl. der Validierung der eingegebenen Formulardaten.*
Tools:	*Visual Studio 2022*
Dauer:	*~30min*

1 ViewModel und Validierungs-Regeln erstellen

Erstellen Sie in den Ordner **Models** eine neue Klasse **CreateMovieViewModel.cs**, welche alle notwendigen Datenfelder enthält, um einen neuen Film anlegen zu können. Eine Id-Eigenschaft ist somit nicht notwendig.

Fügen Sie für jedes Property die Attribute `Required, Range, StringLength` und `Display` hinzu sowie sinnvolle Fehlermeldungen.

- Im Model Movie sind Title, Description, Price, PublishedDate, und Genre ein Muss-Feld
- Title wird auf eine Länge von 30 Zeichen begrenz, Description wird auf eine Länge von 100 Zeichen begrenz.
- Price kann Werte zwischen 1 und 100 annehmen, ist vom Typ Currency und hat eine decimal Einschrankung von (18,2).
- Die Property PublishedDate soll in der UI als „Published Date“ ausgegeben werden.

```
public class CreateMovieViewModel
{
    [Required(ErrorMessage = "Bitte Titel eingeben")]
    public string Title { get; set; }

    // ...

}
```

2 Formularansicht anlegen

Erstellen Sie unter **Views/Movies** eine weitere Ansicht Create (**Create.cshtml**). Verwenden Sie die gerade erzeugte **CreateMovieViewModel** als Datenklasse. In dem Razor-Template können Sie sehen, dass das Formular die Daten an die Post-Methode absendet.

```
<form method="post">
    <div class="form-group">
        <label asp-for="Title" class="control-label"></label>
        <input asp-for="Title" class="form-control" />
        <span asp-validation-for="Title" class="text-danger"></span>
    </div>
    <!-- ... -->
</form>
```

Diese View passen Sie an. So soll der Benutzer aus einer vorhandenen Liste von Genres wählen können. Um ein `Enum` im Formular anzeigen zu lassen können Sie folgenden Razor-Code verwenden:

```
<select asp-for="Genre" class="form-control"
        asp-items="@Html.GetEnumSelectList<MovieGenre>()">
</select>
```

3 Formulardaten im Controller validieren

Achten Sie darauf, dass in der Post-Methode des Controllers der `ModelState` validiert wird. Nur bei erfolgreicher Validierung soll ein neuer Film hinzugefügt werden und dann zur Listenansicht zurück navigiert werden.

Beachten Sie, dass nach einer Fehlereingabe die eingegebenen Formulardaten erhalten bleiben. Das erreichen Sie, indem Sie das Model an den Client zurücksenden.

4 Testen der Funktionalität

Starten Sie die Anwendung und navigieren Sie zu /Movies/Create. Testen Sie dann das Formular:

- Submit ohne Eingaben
- Ungültige Zahlenwerte
- Funktionsweise der Select-Liste
- Korrekte Eingaben

5 Erstellen eines Custom – Attribute (optional)

Erstelle im Verzeichnis Attributes die Klasse ClassicMovieAttribute.cs mit folgender Implementierung:

```
public class ClassicMovieAttribute : ValidationAttribute
{
    public ClassicMovieAttribute(int year) => Year = year;

    public int Year { get; }
    public string GetErrorMessage() =>
        $"Classic movies must have a release year no later than {Year}.";

    protected override ValidationResult IsValid(object value,
        ValidationContext validationContext)
    {
        var movie = (Movie)validationContext.ObjectInstance;
        var releaseYear = ((DateTime)value).Year;

        if (movie.Genre == MovieGenre.Classic && releaseYear > Year)
        {
            return new ValidationResult(GetErrorMessage());

        }
        return ValidationResult.Success;
    }
}
```

Wenden Sie das [ClassicMovie]-Attribute auf die PublishedDate Property an.

6 Weitere Ansicht, um Datenbestand zu aktualisieren (optional)

Als Bonusaufgabe können Sie eine Ansicht Edit und entsprechendes View-Modell anlegen, um den Datenbestand aktualisieren zu können.