# Exercises: MVC Architecture Components

Problems for exercises and homework for the [“C# MVC Frameworks - ASP.NET” course @ SoftUni](https://softuni.bg/courses/asp-net-mvc).

Now let’s implement some of the real, library functionality.

## Display / Editor Templates

Use display templates and editor templates wherever you see fit.

We usually use display templates when we need to reuse the logic, especially for collections. For example, the table at the front page can be done really easily with a display template. The same goes for the “search results” list items.

Editor templates are commonly tied to forms.

You can also use partial views if you need.

## Form Validation

Add validation to all binding models. Don’t allow saving of invalid data. Don’t forget to check whether the **ModelState** is valid where needed.

Add **asp-validation-for** attributes.

Add client-side validation. To do this, have a look at **/Pages/Views/\_ValidationScriptsPartial.cshtml**. This file contains references to the scripts you’ll need.

After adding these to the Layout page (search how to render a partial view), you’ll observe that the client-side validation starts working magically :). Even better, if you don’t forget checking the **ModelState**, the validation will work even if the client has disabled JavaScript.

## Custom Validation

Write a custom validator for the start and end dates of a book borrowing. You have two options. Either:

* Implement **IValidatableObject**, or
* Write your own **ValidationAttribute**s

The second method is usually preferred. The rules are like this:

* Start date
  + Must be filled in
  + Can be any date after 1 January 2000
* End date
  + Optional
  + If filled in, must be after the start date. The same values for start and end date (i.e. borrowing for less than a day) are not allowed

If one borrows a book for a given period, no other period should coincide with it, either fully, or in part.

For example, if a book has been borrowed from 15 July to 20 July, nobody can borrow it from 19 July to 25 July.

A book may be returned before its specified return date. In that case, just update the database with the current date and time.

## Movies

Now that your friends know how generous you are, they start to steal… I mean, borrow, your movies as well. You’ll need to keep track of these as well.

Create the same functionality (movies, directors, borrowers) using the **MVC** pattern.

**Integrate the MVC parts in the same application!**

Of course, one borrower can borrow books and movies; it’s perfectly fine to reuse the same database.

You’ll need a few changes. Next to “Add book”, you’ll need to “Add movie”; the same applies to movie directors. The search results should display movies and directors as well.

Create new forms for adding movies and directors. Create new views for the movie / director details; and movie history (who borrowed this movie, and when).

**If you’ve used templates, partial views, etc. correctly, you’ll be able to reuse most of the logic that you’ve already written. If this isn’t the case, try refactoring your code.**

The key takeouts from this exercise are:

* Both Pages and MVC have the same underlying architecture. It’s up to the developer to decide what’s better, faster, and easier to write
* If you construct our app properly, you should be able to reuse a lot of the “business” logic
  + Actually, this is possible, even if you don’t use ASP.NET at all – simply take out the “business logic” (“service layer”) in another project. We’ll talk about this in a while
* The client should feel absolutely no difference whether you’ve used Pages or MVC