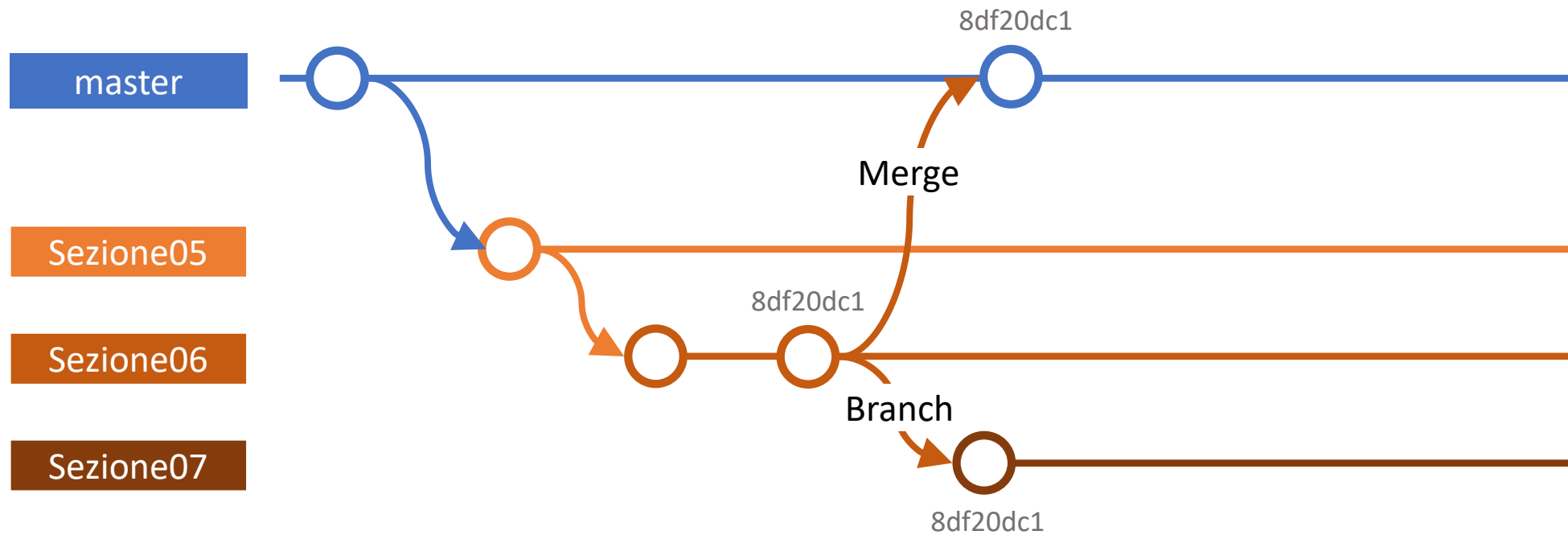


Sezione 07

ASP.NET Core MVC: le View

Git: branch/merge



Creiamo una view Razor

- Contiene codice HTML e C#.

```
<section>  
    <h1>Elenco dei corsi</h1>  
    <ul>  
        <li>Corso 1</li>  
        <li>Corso 2</li>  
        <li>Corso 3</li>  
    </ul>  
</section>
```

Creiamo una view Razor

- Contiene codice HTML e C#.

```
<section>
  <h1>Elenco dei corsi</h1>
  <ul>
    @for(int i = 1; i <= 3; i++)
    {
      <li>Corso @i</li>
    }
  </ul>
</section>
```

Markup HTML

C#

The diagram illustrates the structure of a Razor view. The code is color-coded: HTML tags are brown, and C# code is blue. Arrows point from the labels 'Markup HTML' and 'C#' to their respective code segments. The HTML part includes a section tag, a heading, and a list. The C# part includes a loop that iterates from 1 to 3, rendering each iteration as a list item with the current value of i.

ASP.NET Core MVC: suddivisione in directory

Controllers

 CoursesController.cs

Models

 CourseService.cs

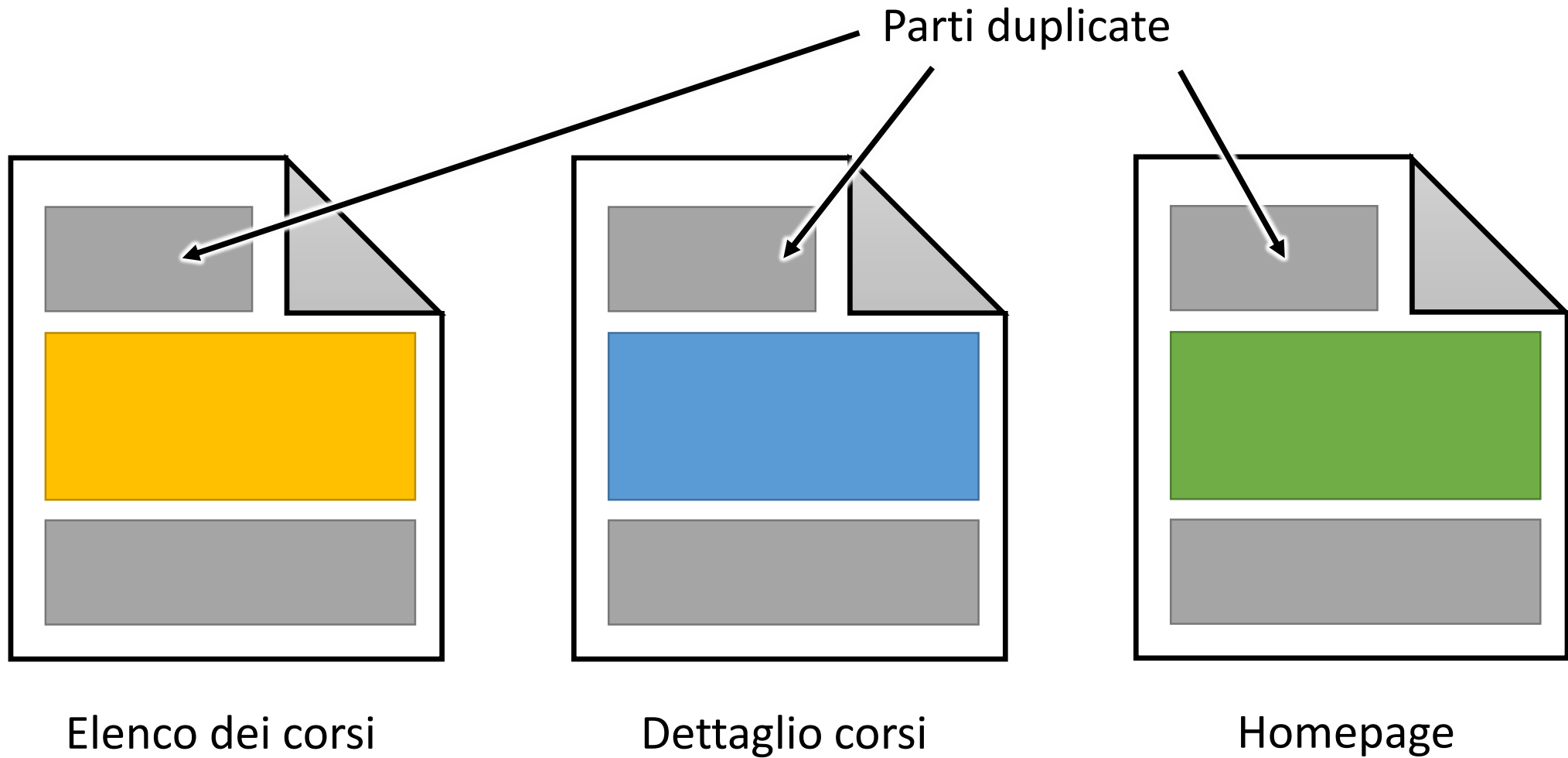
 CourseDetailViewModel.cs

Views

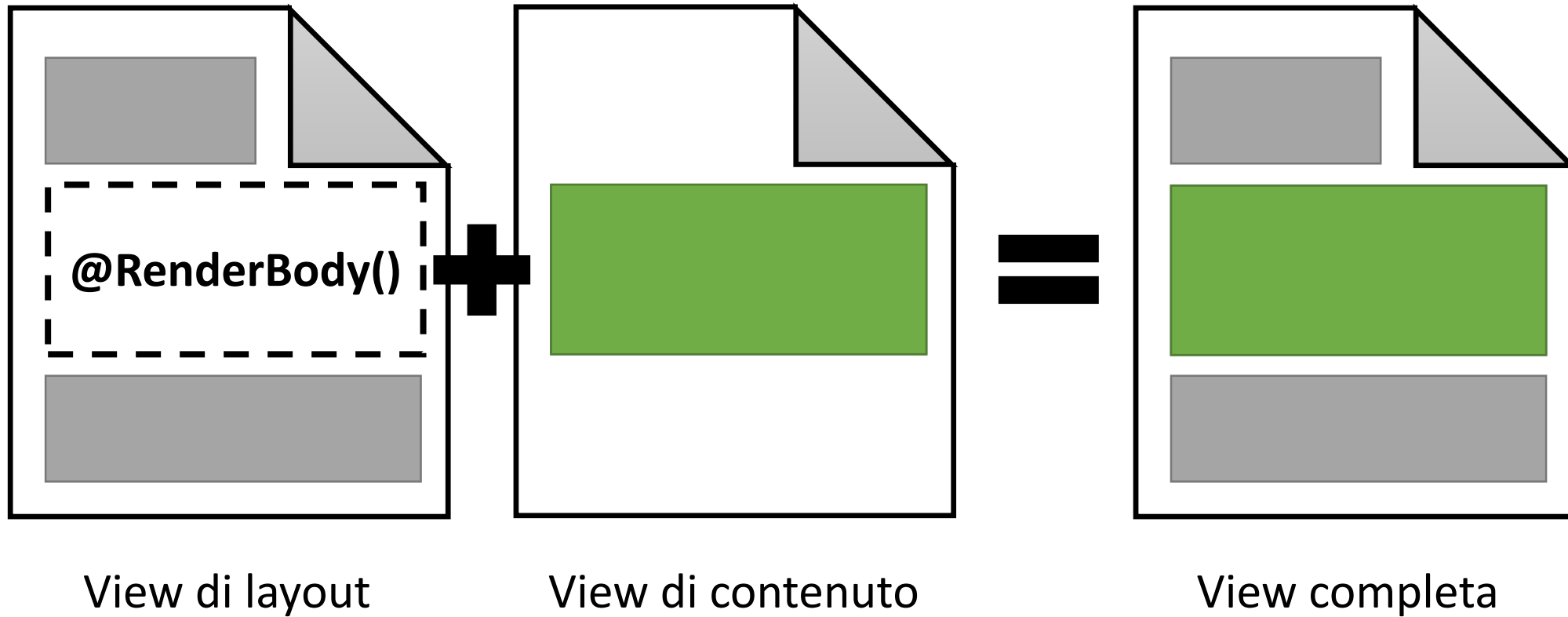
Courses

 Detail.cshtml

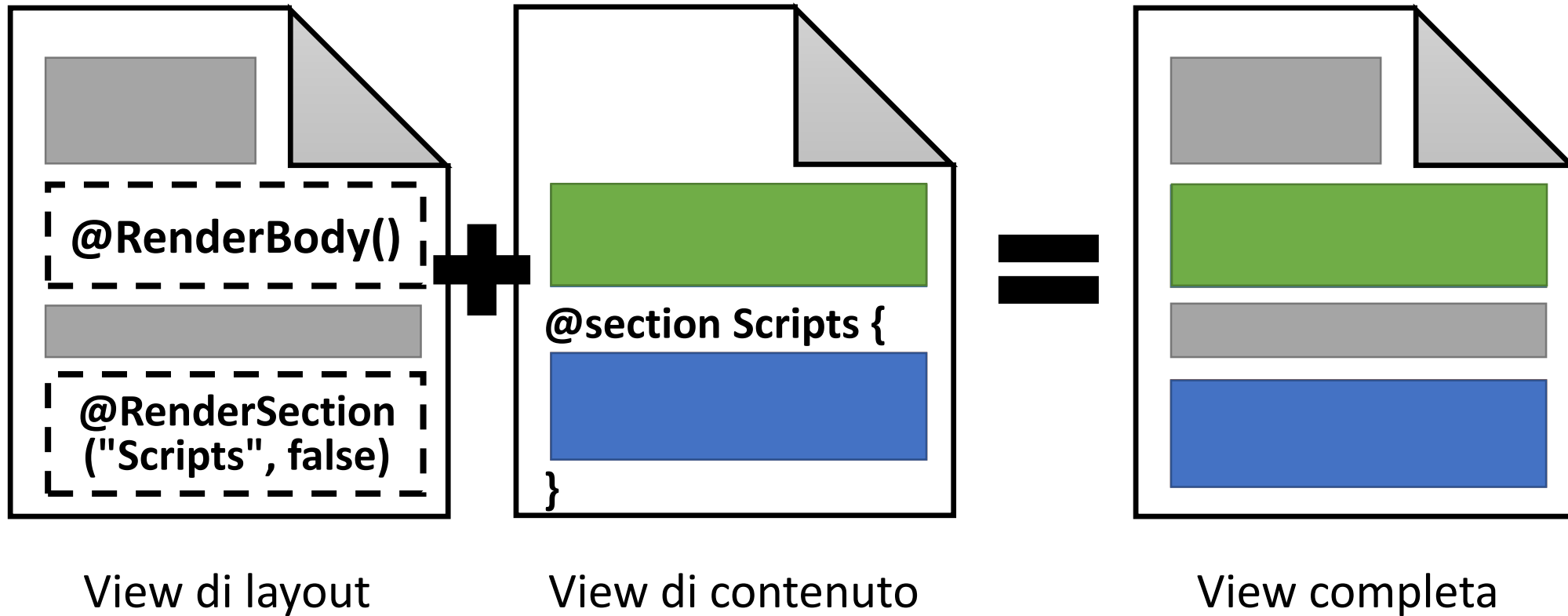
Situazione attuale



View di layout e view di contenuto



View di layout: body e section



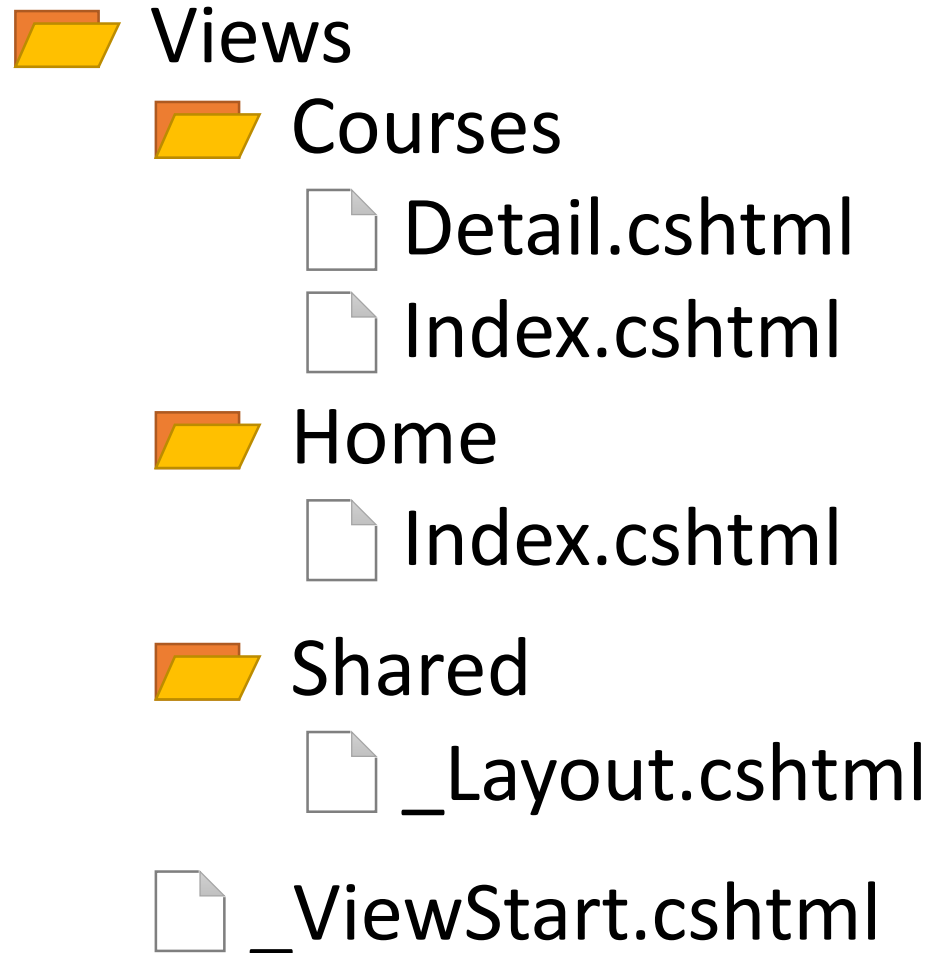
View di contenuto

- Contiene il markup specifico per la pagina corrente;
 - Ad esempio: l'elenco dei corsi;
- Può contenere il markup per le sezioni;
 - In questo caso il markup va circondato da **@section *nome* { ... }**;
- Lei stessa sceglie la sua view di layout;
 - In cima al file mettiamo `@{ Layout = "/Views/Shared/_Layout.cshtml"; }`

View di layout

- Contiene gli elementi fissi dell'applicazione (header, footer, ...);
 - Nel punto in cui vogliamo inserire il contenuto, usiamo **@RenderBody()**;
- Oltre al corpo principale, può definire delle «sezioni»;
 - Nel punto in cui vogliamo inserire la sezione, usiamo **@RenderSection("nome")**;
- Per convenzione, il nome dovrebbe essere prefissato dall'underscore;
 - E la salviamo in /Views/Shared (es. **/Views/Shared/_Layout.cshtml**);

Suddivisione in directory delle view

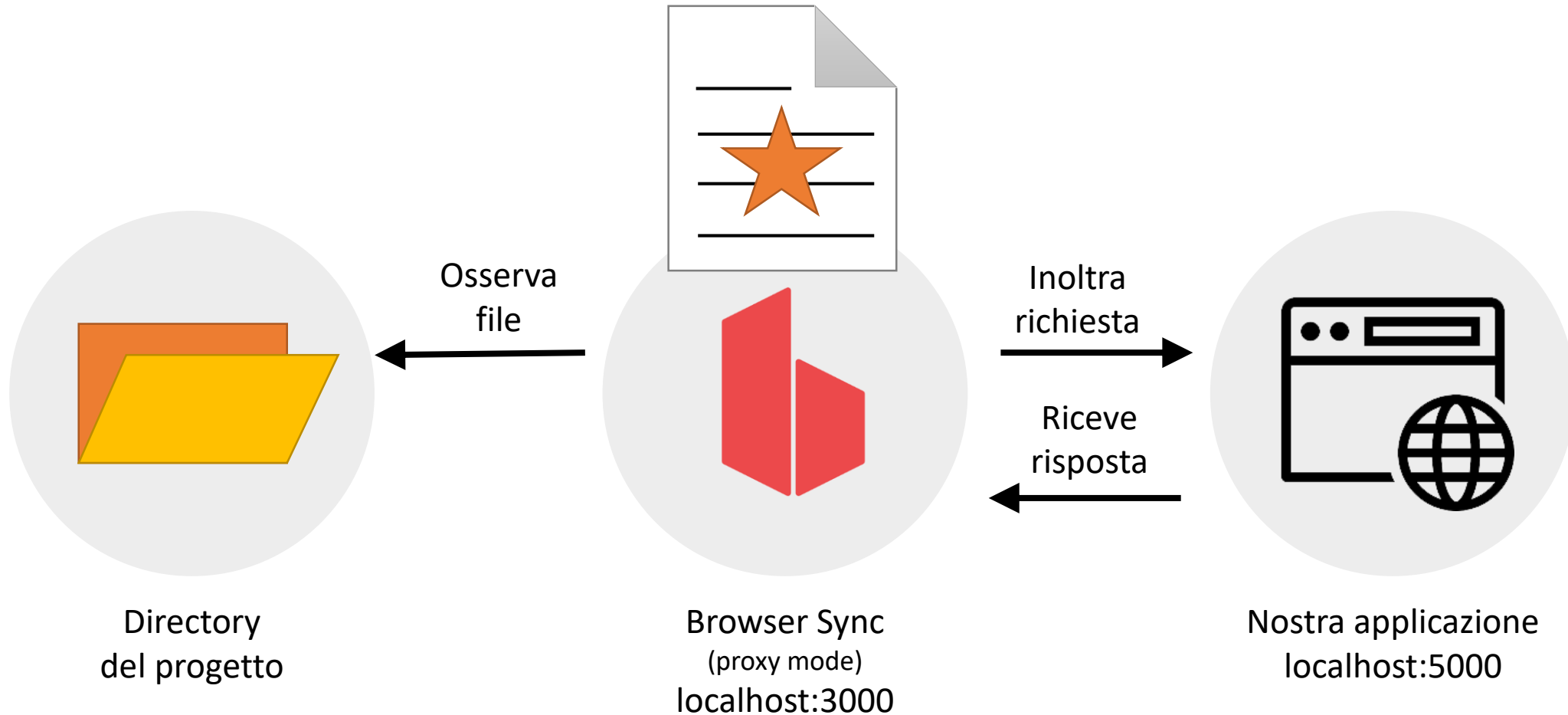


Mettiamo qui il codice da eseguire per ogni view di contenuto.

Ad esempio, per scegliere il layout:

```
@{  
    Layout = "_Layout";  
}
```

Live reload



dotnet watch run

- Permette di ricompilare e riavviare l'applicazione automaticamente ogni volta che si cambia un file di codice C#;
- Il debugger non è disponibile!


Tag helper

- Hanno l'aspetto di normali tag, ad esempio `<a>` o `<environment>`;
- Mantengono leggibile la view perché riducono l'uso di codice C#;
- Possiedono attributi speciali;
- Vengono elaborati dal view engine Razor per produrre HTML valido.

Creare link ad altre pagine

- Normalmente usiamo il tag `<a>` in questo modo.

```
<a href="/Courses/Detail/5">Corso 5</a>
```



```
{controller=Home}/{action=Index}/{id?}
```

...ma se un giorno dovessimo modificare il route template, dovremmo tornare a modificare ogni attributo href.

Il tag helper <a> (*anchor*)

- Ci aiuta a creare link ad altre action o razor pages fornendo i nomi.

Ad esempio, questo tag helper <a>...

```
<a asp-action="Detail" asp-controller="Courses" asp-route-id="5">Corso 5</a>
```

...produce questo codice HTML.

```
<a href="/Courses/Detail/5">Corso 5</a>
```


Il tag helper <a> (*per Razor Pages*)

- Ci aiuta a creare link ad altre action o razor pages.

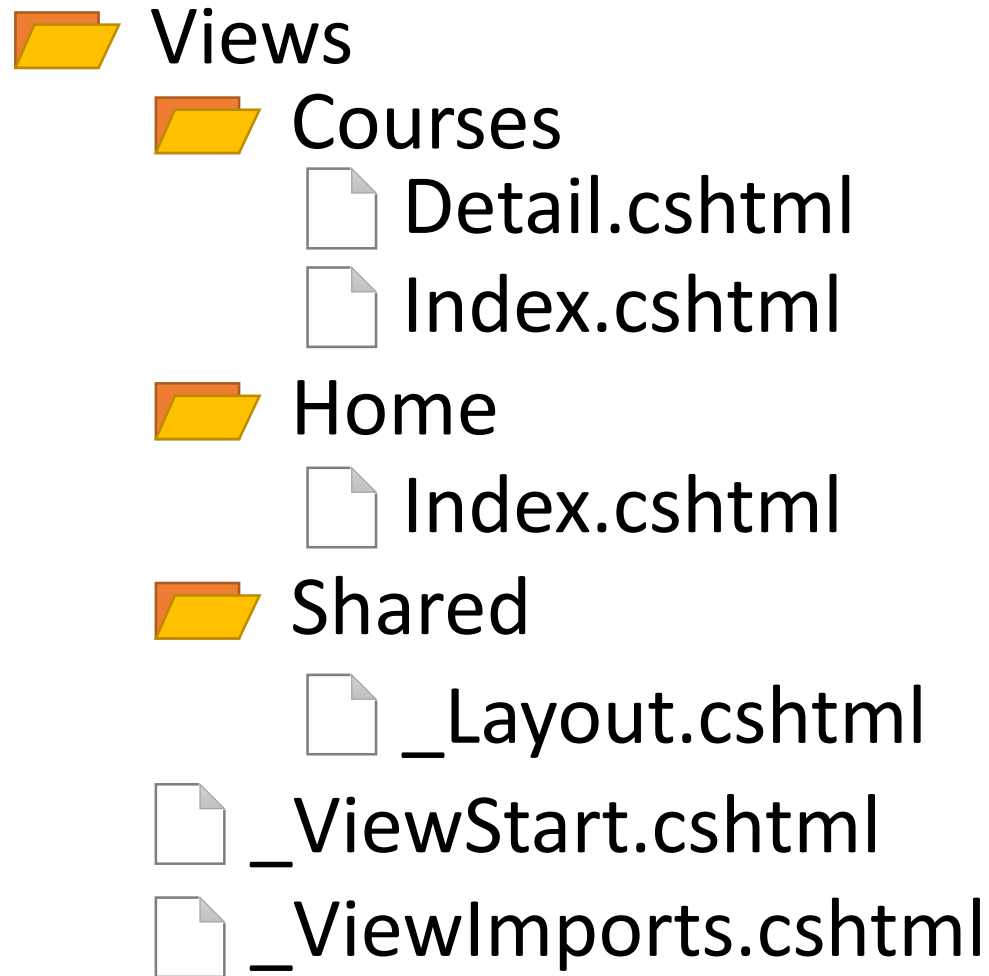
Ad esempio, questo tag helper <a>...

```
<a asp-page="/Courses/Detail" asp-route-id="5">Corso 5</a>
```

...produce questo codice HTML.

```
<a href="/Courses/Detail/5">Corso 5</a>
```

Suddivisione in directory delle view



Aggiungiamo questa riga

```
@addTagHelper *,  
Microsoft.AspNetCore.Mvc.  
TagHelpers
```

Il tag helper <environment>

- Permette di includere/escludere un pezzo di markup HTML in base all'ambiente (es. Development, Production, ...).

Ad esempio, il contenuto di questo tag helper <environment>...

```
<environment include="Development">  
  <script src="/library.js"></script>  
</environment>
```

...apparirà solo in fase di sviluppo.

Il tag helper <environment>

- Permette di includere/escludere un pezzo di markup HTML in base all'ambiente (es. Development, Production, ...).

Ad esempio, il contenuto di questo tag helper <environment>...

```
<environment exclude="Development">  
  <script src="/library.js"></script>  
</environment>
```

...NON apparirà in fase di sviluppo ma in tutti gli altri ambienti sì.

Pagina di elenco dei corsi

2. Per ogni corso in elenco visualizzare un titolo, un'immagine rappresentativa, l'autore, la valutazione, il prezzo intero e il prezzo corrente di acquisto;
5. Cliccando uno dei risultati nell'elenco, si accede alla pagina di dettaglio del corso.

The diagram illustrates a course listing item with the following components and annotations:

- Immagine**: Points to a thumbnail image of a course titled "ONLINE MARKETING".
- Autore**: Points to the author's name, "di Mario Rossi".
- Titolo**: Points to the course title, "Web marketing facile".
- Valutazione**: Points to a five-star rating.
- Prezzo corrente**: Points to the current price, "EUR 17.99".
- Prezzo intero**: Points to the original price, "EUR 19.99", which is shown in a lighter gray font.
- Al dettaglio**: Points to a blue button labeled "Dettagli", which is highlighted with a blue border.

The course listing is presented in a horizontal layout with a light gray background and a thin horizontal line separating it from the next item.

Visualizzazione responsive

Usiamo il grid system di Bootstrap per far fluire i corsi l'uno sotto l'altro su smartphone e tablet.

