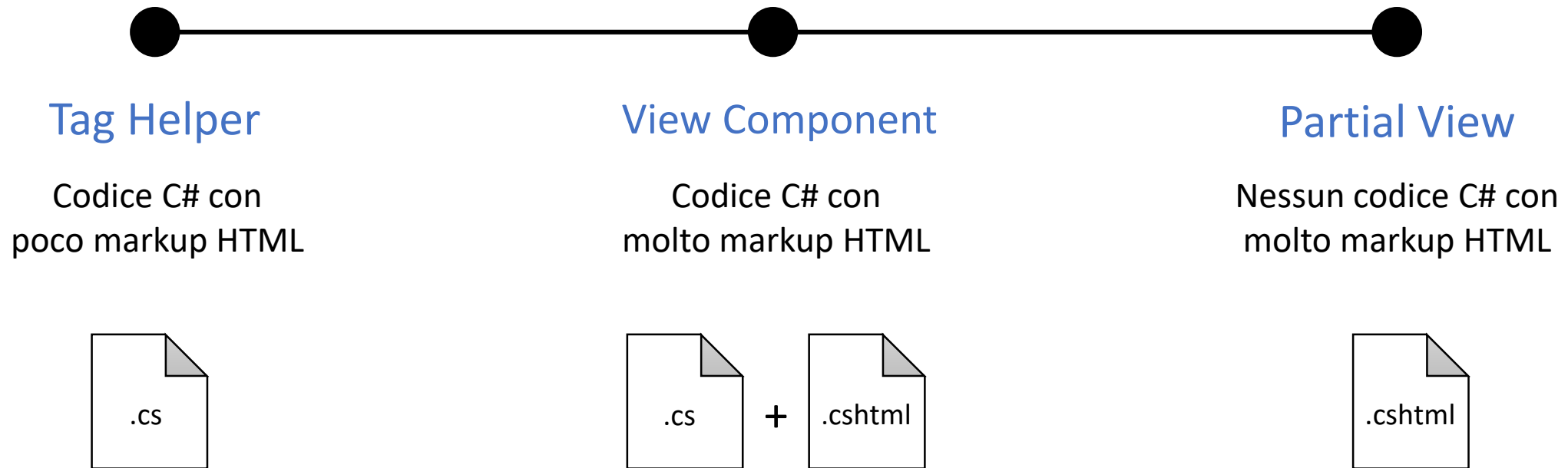


Sezione 14

Organizzare la UI in componenti

Organizzare la UI in componenti

I componenti promuovono il riutilizzo di codice nelle view.






Organizzare la UI in componenti

Diagram illustrating the organization of a UI into components, using a book listing table as an example.

The table displays book information, including the title, author, rating, price, and a "Dettaglio" (Details) button.

Handwritten annotations highlight specific components:

- Tag Helper:** Points to the "Valutazione" (Rating) dropdown menu.
- Partial View:** Points to the entire row of the table, indicating it is a reusable component.
- View Component:** Points to the pagination controls (1, 2, 3, 4, >).

Titolo	Valutazione	Prezzo	
 Web marketing facile <i>di Adelinda Greco</i>	★★★★☆	EUR 17.99 EUR 19.99	Dettaglio
 L'ABC del fai da te <i>di Salvatore Marino</i>	★★★★☆	EUR 7.99	Dettaglio
 Come fare gli origami <i>di Nella Pisano</i>	★★★★☆	EUR 30.99 EUR 34.99	Dettaglio

1 2 3 4 >

Creare tag helper personalizzati

```
[HtmlTargetElement("stars")]  
public class RatingTagHelper : TagHelper  
{  
    public double Value { get; set; }  
  
    public override void Process(TagHelperContext context, TagHelperOutput output)  
    {  
        output.TagName = "span";  
        output.Content.AppendHtml("...");  
    }  
}
```

`<stars value="@Model.Rating"></stars>`

Creare tag helper personalizzati

Ricorda: registra i tag helper personalizzati mettendo una direttiva `@addTagHelper` nella view `/Views/Shared/_ViewImports.cshtml`

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

```
@addTagHelper *, MyCourse
```

```
@using System.Collections.Generic
```

```
@using MyCourse.Models.ViewModels
```

*Nome dell'assembly
dell'applicazione*



Case styles

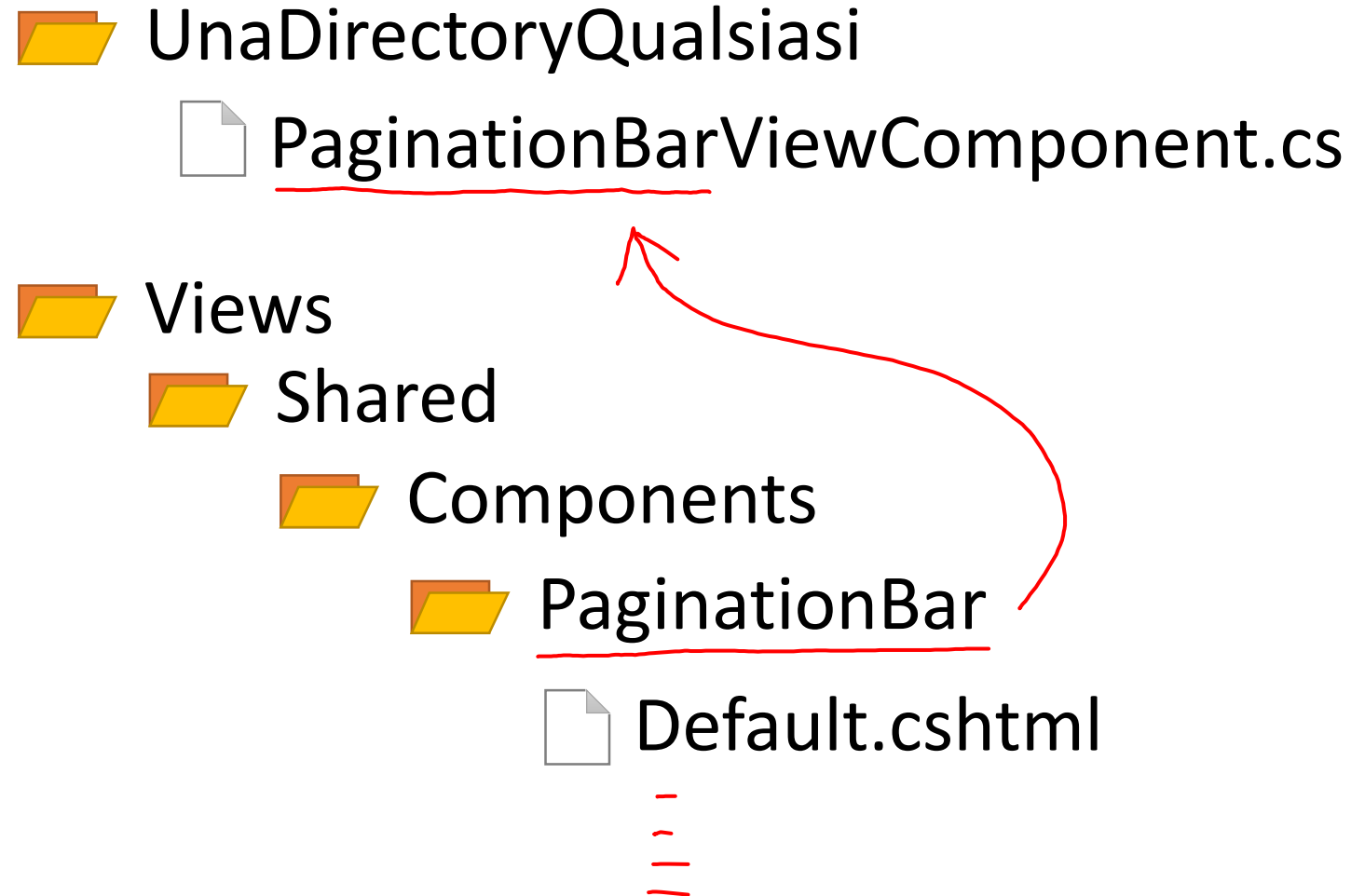
PascalCase

camelCase

kebab-case

snake_case

Creare un view component



Create view component

```
[ViewComponent(Name = "Pagination")]  
public class PaginationBarViewComponent : ViewComponent  
{  
    public IViewComponentResult Invoke(CourseListViewModel model)  
    {  
        return View(model);  
    }  
}
```

<vc:pagination model="@Model"></vc:pagination>



Creare view component

Ricorda: registra i view component mettendo una direttiva `@addTagHelper` nella view `/Views/Shared/_ViewImports.cshtml`

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

```
@addTagHelper *, MyCourse
```

```
@using System.Collections.Generic
```

```
@using MyCourse.Models.ViewModels
```

*Nome dell'assembly
dell'applicazione*



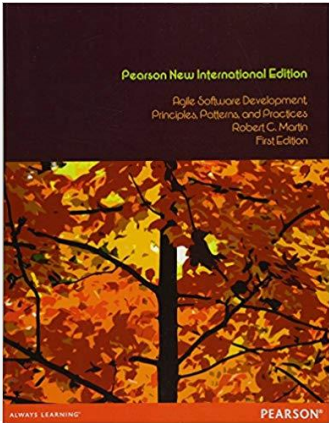
Accoppiamento debole



"Program to an interface, not an implementation"

Da "Design Patterns: elements of reusable object-oriented software"

Principio di segregazione delle interfacce



"No client should be forced to depend on methods it does not use."

Da "Agile Software Development: Principles, Patterns, and Practices"

Robert "Uncle Bob" Martin
cleancoders.com



MENO C'È

MENO SI ROMPE

Principi SOLID

Single responsibility

Ogni componente dovrebbe avere una singola responsabilità.

Open / Closed

Ogni componente può aprirsi ad aggiunte ma restare chiuso a modifiche al suo comportamento.

Liskov substitution principle

Le dipendenze devono poter essere sostituite con altre derivate da esse.

Interface segregation

Un componente non dovrebbe dipendere da membri che non usa.

Dependency inversion

Ogni componente dovrebbe avere una singola responsabilità

Creare una partial view



Views

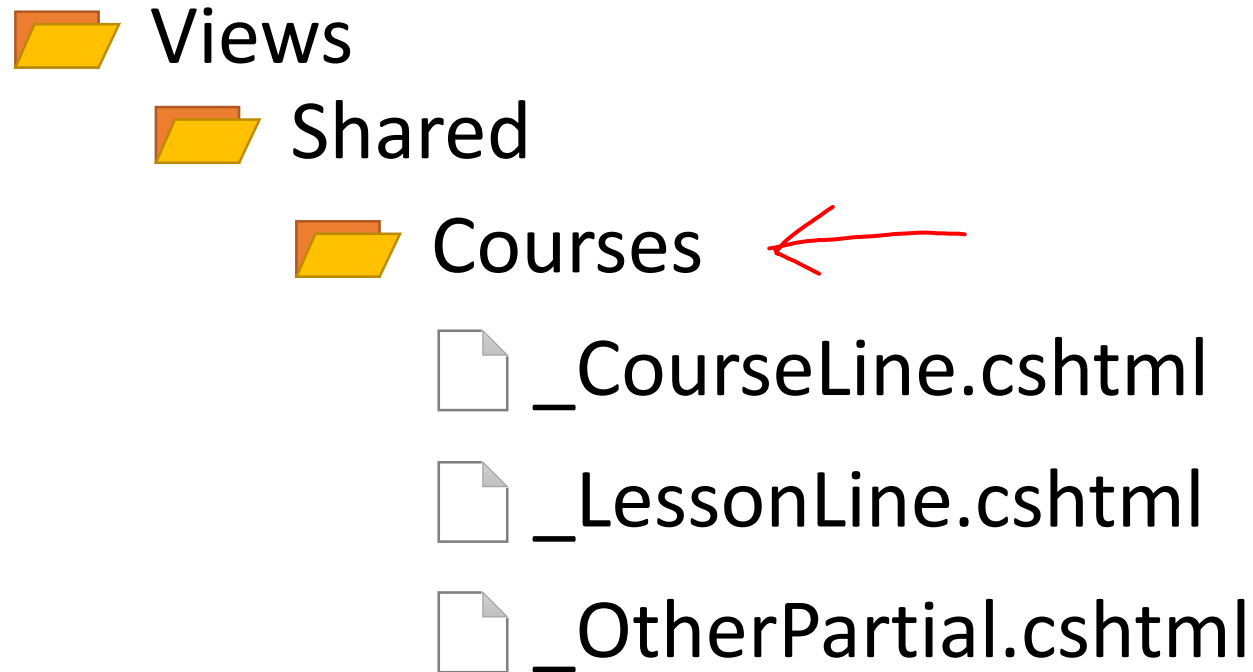


Shared



_CourseLine.cshtml

Organizzare le partial view in directory



Creare partial view

```
@model CourseViewModel  
<!-- Codice C# / HTML qui -->
```



```
<partial name="Courses/_CourseLine" model="@course"></partial>
```

Organizzate le view dal generale al particolare, come se doveste organizzare un cassetto



