



**Eötvös Loránd Tudományegyetem
Informatikai Kar**

Webes alkalmazások fejlesztése

3. előadás

Megjelenítés és tartalomkezelés (ASP.NET Core)

Cserép Máté

mcserep@inf.elte.hu

<http://mcserep.web.elte.hu>

Megjelenítés és tartalomkezelés

Nézetek kezelése

- Sok esetben a nézetünk különböző részekből áll, amelyek egymástól függetlenül változhatnak
 - bizonyos részek (pl. címsor, menü) több oldalon is szerepelnek, másokat folyamatosan cserélünk
 - az ismétlődő részek adják meg a weblapunk egységes kinézetét
- Az ismétlődő tartalmat kiemelhetjük, és felhasználhatjuk több nézetben
 - ezek így nem feltétlenül egy vezérlőhöz tartoznak, hanem *megosztottak* a vezérlők között (*shared view*), amelyeket a **Views/Shared** könyvtárba helyezünk

Megjelenítés és tartalomkezelés

Parciális nézetek

- A *parciális nézet* (*partial view*) olyan nézet, amely nem a teljes oldalt, csak annak egy részét adja meg
 - ezt a tartalmat egy másik nézetben megjeleníthetjük a `Html.RenderPartial` utasítással
 - megadjuk a nézet nevét, emellett megadhatjuk az ott használandó modellt, illetve nézet tulajdonságokat

- pl. (`Item.cshtml`):

```
@model String @* a modell szöveg típusú *@
```

```
<span><b>@Model</b> @* nem teljes a tartalom *@  
    @Html.ActionLink("See details", "Details",  
                    new { param = Model });
```

```
</span>
```

Megjelenítés és tartalomkezelés

Parciális nézetek

- pl. (MyView.cshtml):

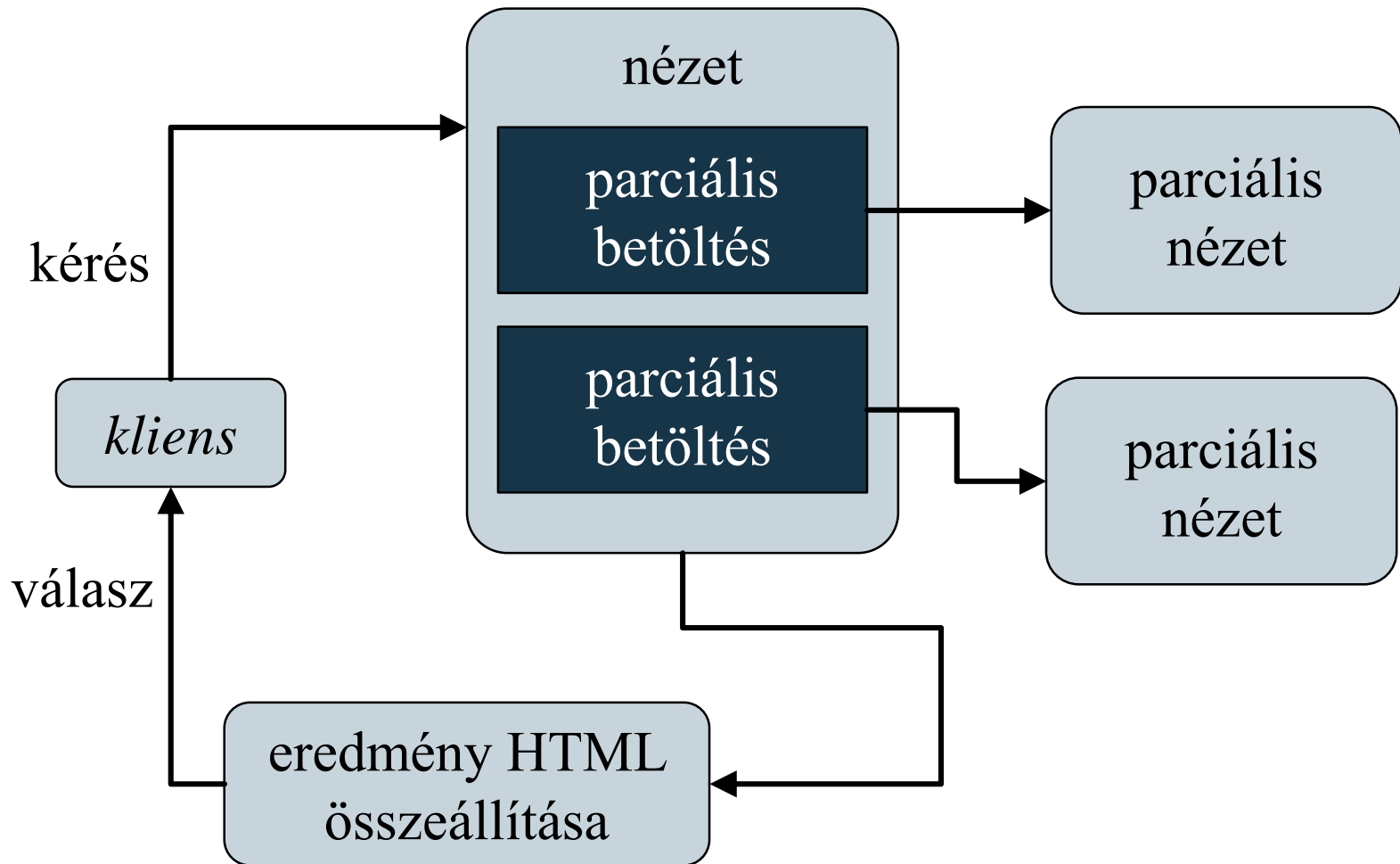
```
@model IEnumerable<String>

...
<body>
  <div>
    @foreach (String name in Model) {
      Html.RenderPartial("Item", name);
      // meghívjuk a parciális nézetet,
      // átadjuk a nézetmodellt
    }
  </div>
</body>

...
```

Megjelenítés és tartalomkezelés

Parciális nézetek



Megjelenítés és tartalomkezelés

Parciális nézetek

- a parciális nézet közvetlenül is létrehozható egy vezérlőből a **PartialView** metódussal, ekkor a nézetben a **RenderAction** művelet fogja a tartalmat betölteni
 - így a modellt, és a nézet tulajdonságait a vezérlő fogja definiálni

- pl.:

```
public class ProductController : Controller {  
    ...  
    public PartialViewResult Details(int id) {  
        Item item = ...  
        return PartialView("Details", item);  
    }  
}
```

Megjelenítés és tartalomkezelés

Elrendezések

- Az *elrendezés (layout)* lehetőséget ad, hogy egy oldalon belül több cserélhető tartalmat adjuk meg, amelyeket más nézetekből töltünk be
 - az *elrendező nézet* a keret, amely az állandó tartalmat definiálja
 - a behelyettesíthető tartalmak a szakaszok (*section*), amelyek az egyes nézetekben definiáltak
 - a szakaszokat `@section <név> { ... }` blokk segítségével adjuk meg
 - speciális szakasz a törzs (*body*), amelyet nem jelölünk
 - be kell hivatkoznunk az elrendezést a **Layout** tulajdonsággal

Megjelenítés és tartalomkezelés

Elrendezések

- pl. (View.cshtml):

```
@model ProductsModel
```

```
@{
```

```
    Layout = "~/Views/Shared/_Layout.cshtml;
```

```
    // használunk egy elrendezést
```

```
}
```

```
@section mainMenu { // szakasz a menühöz
```

```
    foreach (String item in Model.ItemTypes)
```

```
        <div>@Html.ActionLink(...)</div>
```

```
}
```

```
@* a közvetlenül megadott tartalom a törzs *@
```

```
<div id="title">List of Products</div>
```

```
...
```

```
@section mainFooter { ... } // újabb szakasz
```


Megjelenítés és tartalomkezelés

Elrendezések

- az elrendező nézetben a törzset a **RenderBody()**, a további szakaszokat a **RenderSection(<név>)** utasítással helyezhetjük el
 - amennyiben nem kötelező, hogy egy szakasz definiált legyen, a **required** opció hamis értékével ezt jelölhetjük
 - a szakasz megléte ellenőrizhető az **IsSectionDefined(<név>)** művelettel
- az elrendező nézet fájlnevét konvenció szerint aláhúzással kezdjük, az alapértelmezett elrendezés a **Views/Shared/_Layout.cshtml**
 - ennek használatához nem kell a **Layout** tulajdonság
- az elrendezések egymásba ágyazhatóak

Megjelenítés és tartalomkezelés

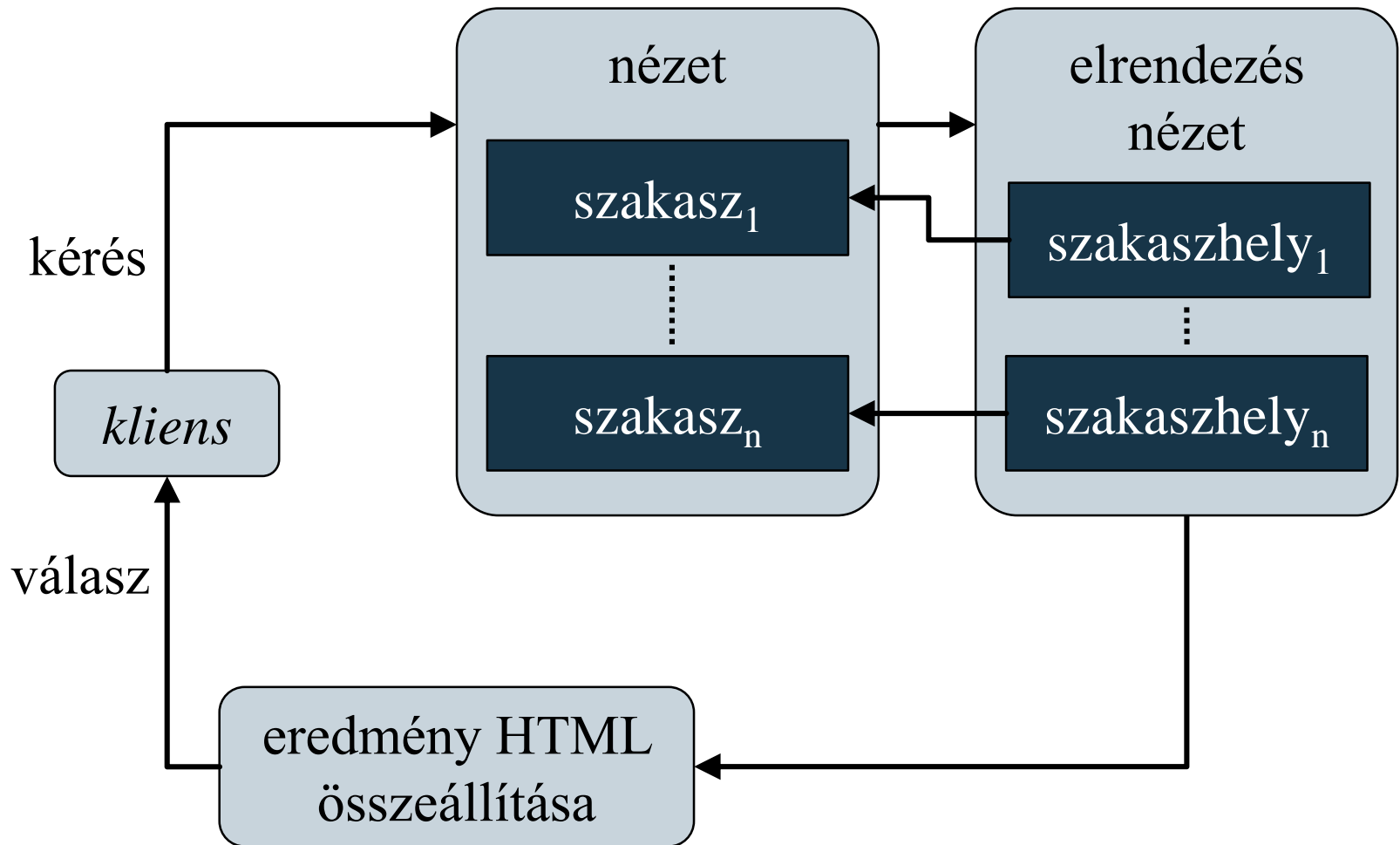
Elrendezések

- pl.:

```
... <body>
  ... @* egyéb tartalom *@
  <div id="menu">
    @RenderSection("mainMenu")
    @* egy szakasz betöltése *@
  </div>
  <div id="mainpage">
    @RenderBody() @* fő szakasz betöltése *@
  </div>
  <div id="footer">
    @RenderSection("mainFooter", false)
  </div> @* opcionális szakasz *@
</body> ...
```

Megjelenítés és tartalomkezelés

Parciális nézetek



Megjelenítés és tartalomkezelés

Nézetek kezelése

- Elrendezéseket célszerű használni, amennyiben:
 - az oldalunk keretét, struktúráját szeretnénk definiálni (pl. menü, fejléc)
 - ugyanazok elemeket szeretnénk ugyanolyan módon megjeleníteni több oldalon
- Parciális nézeteket célszerű használni, amennyiben:
 - ugyanolyan elemeket változó környezetben szeretnénk használni (pl. bejelentkező doboz), vagy egy adott elemet szeretnénk cserélhetővé tenni (pl. táblázat/diagram)
 - az oldalnak csak egy részét szeretnénk változtatni, illetve újra betölteni

Megjelenítés és tartalomkezelés

Fájltartalom kezelése

- Lehetőségünk van tetszőleges fájl tartalmat (pl. képek, dokumentumok, csomagolt fájlok) küldeni a felhasználónak
 - a tartalom rendelkezik egy típussal (*internet media type*), amennyiben a böngésző meg tudja jeleníteni, akkor megjelenítheti, egyébként felkínálhatja letöltésre
 - fájl tartalmat a **File** metódussal tölthetünk be
 - megadhatjuk a tartalmat binárisan, adatfolyamként, vagy elérési útvonallal
 - meg kell adnunk a típust
 - megadhatjuk a letöltési fájl nevet (ekkor mindenképpen letöltésre ajánlja fel)

Megjelenítés és tartalomkezelés

Fájltartalom kezelése

- pl.:

```
return File("/Content/Pictures/image.jpg",
           "image/jpg");
// JPEG kép betöltése a fájlrendszeről
...
Byte[] imageFile = ... // betöltés adatbázisból
Return File(imageFile, "image/png",
           "image.png");
// PNG kép betöltése az adatbázisból, majd
// letöltése (alapértelmezetten) image.png
// néven
```
- a megjelenített fájl tartalmát az `Url.Action` művelet használatával beágyazhatjuk a nézetbe

Megjelenítés és tartalomkezelés

Példa

Feladat: Valósítsuk meg egy utazási ügynökség weblapját, amelyben apartmanok között böngészhetünk.

- a kódismétlődés elkerülése végett használjuk elrendezést (`_Layout.cshtml`), amelyben megadjuk a fejléctet, illetve a városok listáját
- az **Index** és **Details** oldalakban csak a különböző részeket adjuk meg
- a képek megjelenítéséhez két új akciót adunk meg, egyikkel a listában tudunk minden épülethez egy képet adni (**ImageForBuilding**), a másikkal egyenként tudjuk a képeket betölteni kis és nagy méretben (**Image**)
 - előfordulhat, hogy nincs kép (**NoImage.png**)

Objektumrelációs adatkezelés

Példa

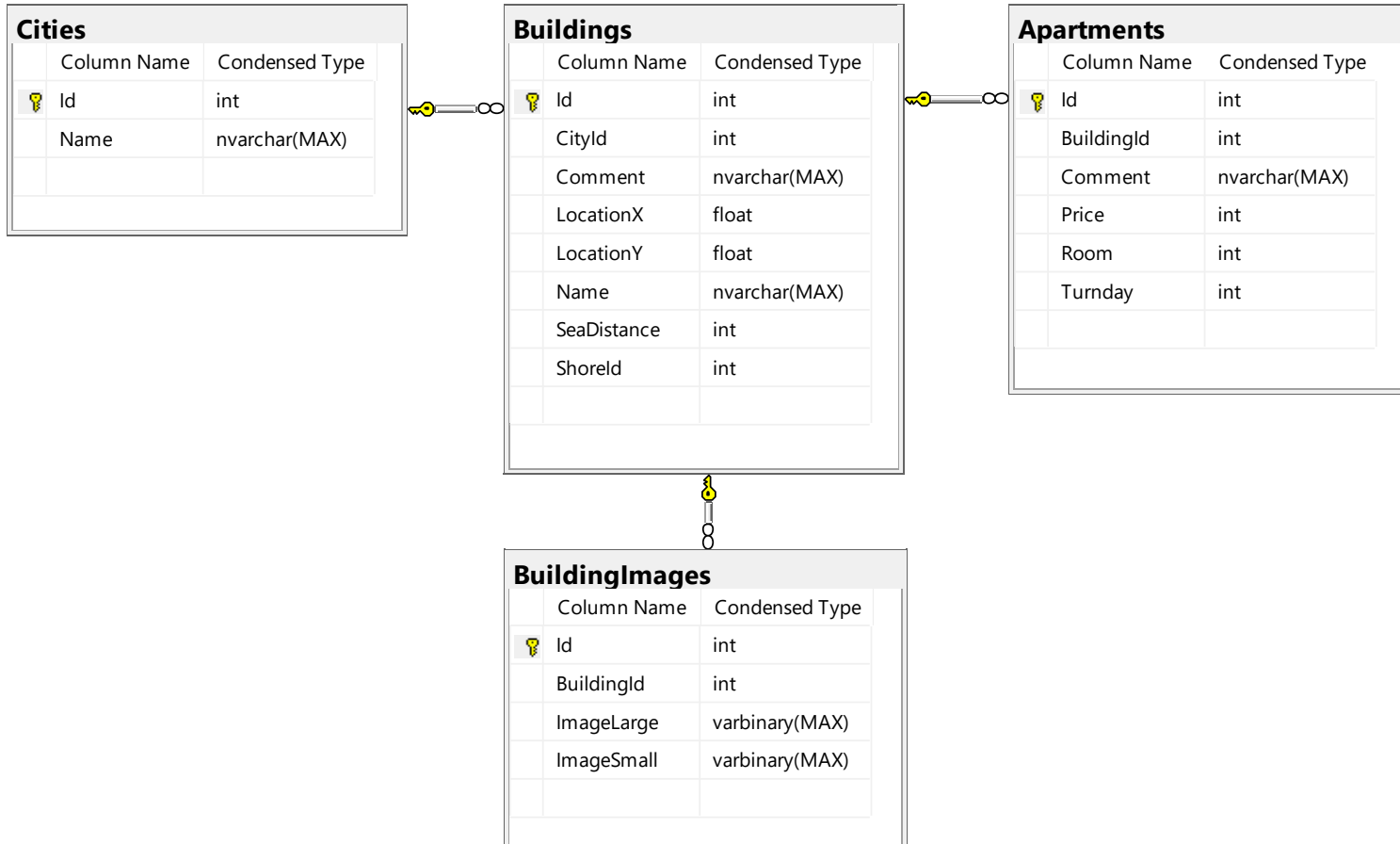
Tervezés (entitás modell):

- a **City** entitás tárolja a városok adatait tartalmazza
- a **Building** entitás az épületek adatait tartalmazza, benne a város azonosítójával
- az **Apartment** entitás az apartman adatokat tárolja, benne az épület azonosítójával
- a **BuildingImage** entitás tárolja az épületek bemutató képeit, minden képből egy nagyobb, és egy kisebb változatot, valamint az épület azonosítóját

Objektumrelációs adatkezelés

Példa

Tervezés (adatbázis):



Megjelenítés és tartalomkezelés

Példa

- az épületek elhelyezkedése tárolva van az épületek táblában (**LocationX**, **LocationY**), ezt csak át kell adnunk szöveges formában a térképnek (ügyelve arra, hogy ponttal kell elválasztani a tizedesrészt)
- a Google térképet a **Details** oldalon jelenítjük meg, megadva a méretet, illetve a megjelölt koordinátát. A megjelenítéshez a [*Google Maps JavaScript API-t*](#) használjuk.
- a konfigurációban (**appsettings.json**) megadjuk az egyedi térképkulcsot, amely a következő helyről igényelhető: [*http://console.developers.google.com*](http://console.developers.google.com)