Eötvös Loránd Tudományegyetem
Informatikai Kar

Webes alkalmazások fejlesztése

2. fejezet

Webfejlesztés MVC architektúrában (ASP.NET)

Giachetta Roberto

A jegyzet az ELTE Informatikai Karának 2016. évi jegyzetpályázatának támogatásával készült

Webfejlesztés MVC architektúrában

Fejlesztés ASP.NET alapon

- A *Microsoft ASP.NET* az *ASP* (*Active Server Pages*) továbbfejlesztése .NET programozás támogatással
  - egy (szerver oldali) programozási felület dinamikus weblapok készítésére HTTP protokollon keresztül
  - három programozási modellt kínál:
    - *Web Forms*: az alkalmazás dinamikus részeit vezérlők és eseménykezelés segítségével valósítja meg
    - *Web Pages*: egyszerű, jórészt statikus alkalmazások megvalósítása, amelyek támogatnak dinamikus funkciókat és adatkezelést
    - *MVC*: összetett weblapok fejlesztésére szánt modell

Webfejlesztés MVC architektúrában

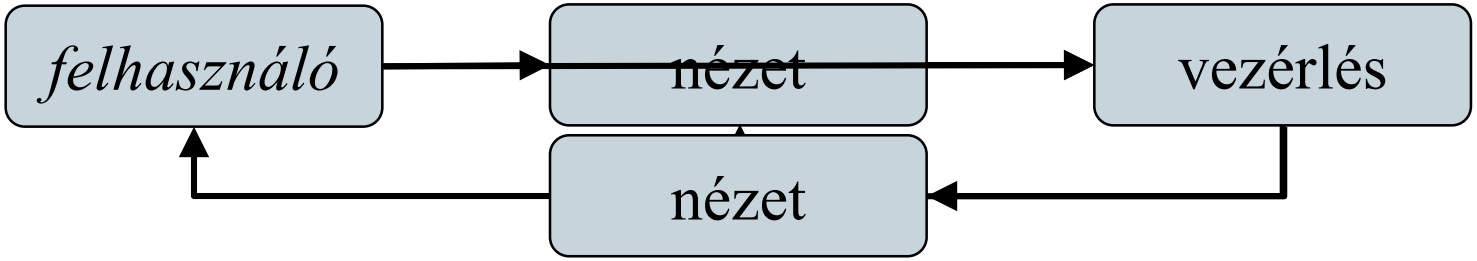
Szoftver architektúrák

- A *modell/nézet* (MV) architektúra elválasztja a háttérben végzett tevékenységeket a felhasználói felülettől és interakciótól
- A *modell/nézet/nézetmodell* (MVVM) architektúra leválasztja a felhasználói interakció kezelését, valamint az adatmegjelenítést a felülettől
  - a nézetmodell átalakítja az adatok a megfelelő megjelentéshez, tehát átjáróként (proxy) szolgál
  - az utasítások nem eseménykezelők, hanem parancsok (command) formájában jelennek meg
  - a vezérlés (*control*) külön programegységben történik

Webfejlesztés MVC architektúrában

Szoftver architektúrák

- Asztali környezetben a felhasználó a nézettel teremt kapcsolatot, amely biztosítja a megfelelő utasítás végrehajtását (eseménykezelő, parancs)
- Webes környezetben a felhasználó az adott erőforrással teremt kapcsolatot, amit elsősorban az útvonala határoz meg
  - vagyis a felhasználó közvetlenül a vezérlést veszi igénybe
  - a vezérlésre az alkalmazásnak egy (új) nézettel kell válaszolnia, ami az adott erőforráshoz tartozik



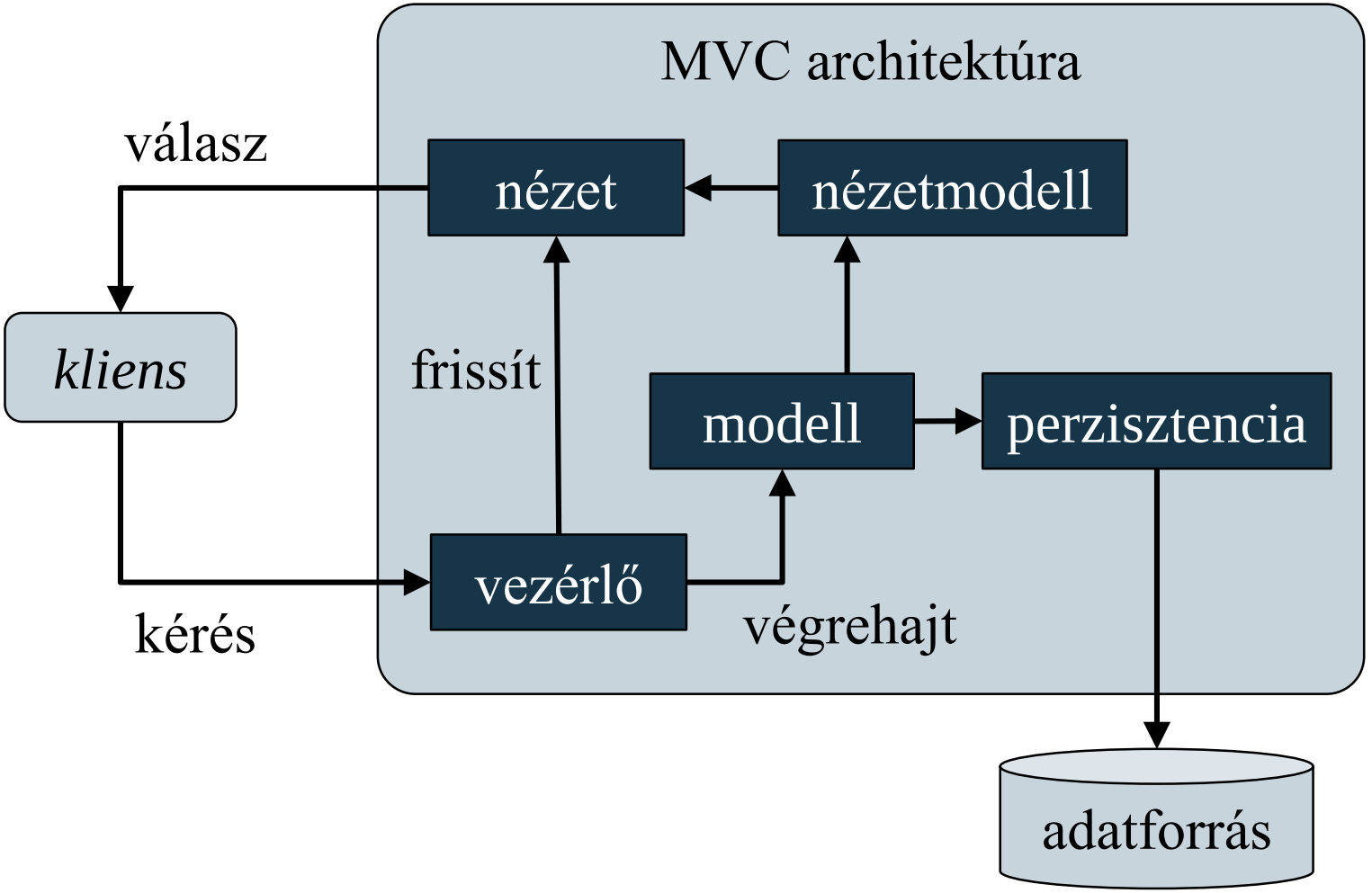
Webfejlesztés MVC architektúrában

Az MVC architektúra

- A *modell/nézet/vezérlő* (*Model-View-Controller*, *MVC*) architektúra egy többrétegű felépítést definiál, amely jól illeszkedik a webes környezethez
  - a *vezérlő* a kérések kiszolgálója, amely biztosítja a nézetet a kérés eredménye alapján
  - a *nézet* a felület (jórészt deklaratív) definíciója, nem tartalmaz háttérkódot, csupán az adatokat kéri a modelltől
  - a *modell* a logikai funkciók végrehajtása (üzleti logika)
  - a *nézetmodell* egy átjáró, amely az adatokat a nézet számára megfelelő módon prezentálja
  - a *perzisztencia* felel az adatelérésért

Webfejlesztés MVC architektúrában

Az MVC architektúra



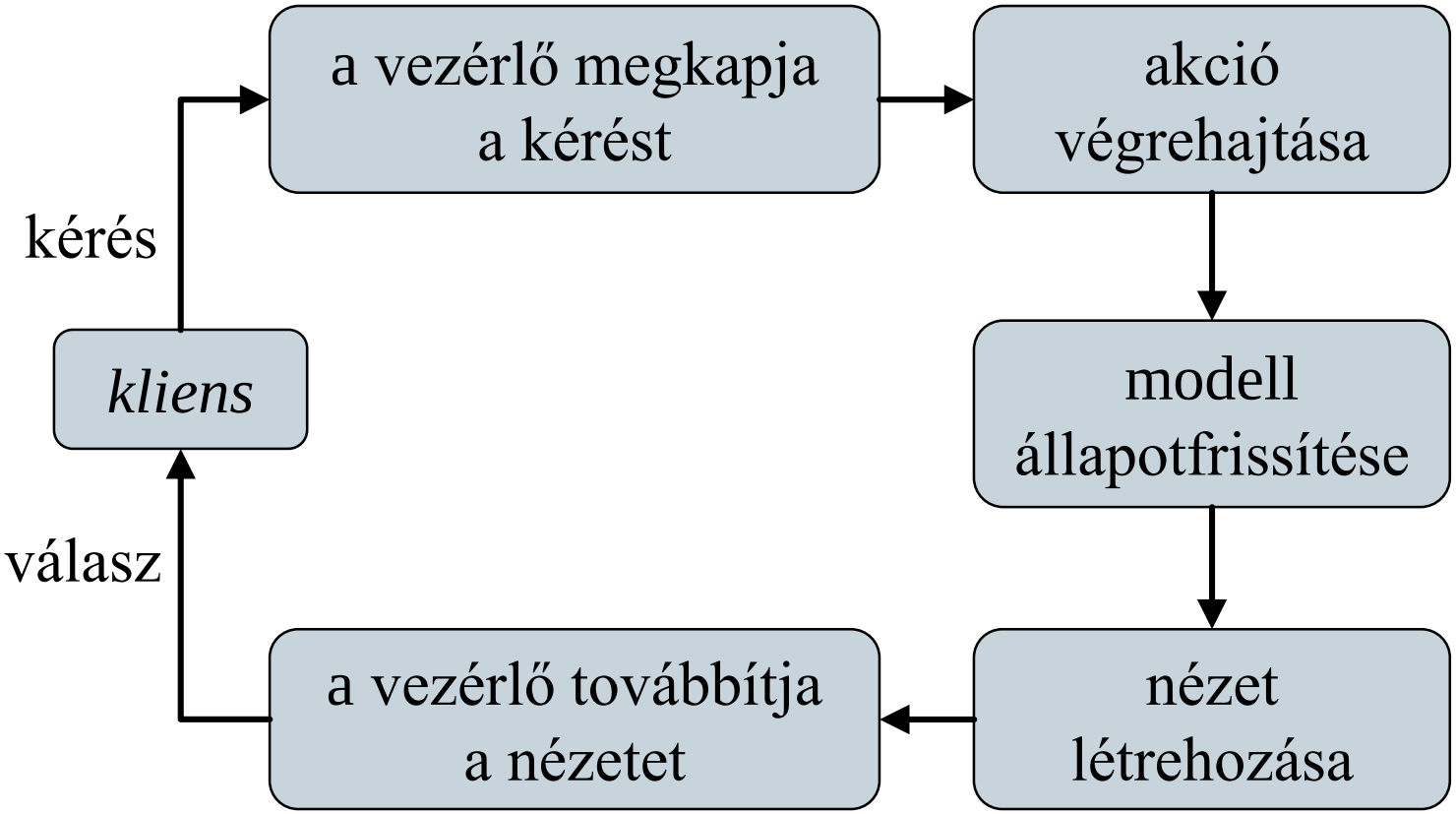
Webfejlesztés MVC architektúrában

Az MVC architektúra

- Az MVC architektúra végrehajtási ciklusa:
  1. a felhasználó egy kérést ad a szervernek
  2. a vezérlő fogadja a kérést, majd a modellben végrehajtja a megfelelő akciót (*action method*)
  3. a modellben végrehajtott akció állapotváltást okoz
  4. a vezérlő begyűjti az akció eredményét (*action result*), majd létrehozza az új nézetet (*push-based*)
     - egy másik megközelítés, hogy a nézet is lekérdezze a vezérlők eredményeit (*pull-based*)
     - az adatok nézetmodell segítségével kerülnek a nézetbe
  5. a felhasználó megkapja a választ (nézetet)

Webfejlesztés MVC architektúrában

Az MVC architektúra



Webfejlesztés MVC architektúrában

MVC alkalmazások elemei

- Az ASP.NET MVC alkalmazások az MVC architektúrát valósítják meg dedikált komponensek segítségével
  - a nézet egy olyan osztály (`View`), amelyet alkalmas leíró nyelv segítségével fogalmazunk meg (ASPX, Razor, …)
    - a nézetben a modell tartalmára hivatkozhatunk (adatkötéssel), illetve használhatunk HTML kódot
  - a vezérlő (`Controller`) a tevékenységeket tartalmazó osztály, amiben akciókat (metódusokat) definiálunk
    - az akció eredménye (`ActionResult`), amely általában egy nézet
  - a modell és a perzisztencia tetszőleges lehet

Webfejlesztés MVC architektúrában

MVC alkalmazások elemei

- Pl. (vezérlő több akcióval):

```
public class MyController : Controller {
    // vezérlő
    public ActionResult Index() { // akció
        return View("Welcome to the website!");
        // az eredmény egy nézet lesz, benne a
        // szöveg, ez lesz a nézetmodell
    }
    public ActionResult List(){
        … // adatok elkérése a modellből
        return View(list);
    }
    public ActionResult Details(String id){ … }
    …
```

Webfejlesztés MVC architektúrában

MVC alkalmazások elemei

- Pl. (Razor nézet):

```
@model String @* a modell típusa szöveg lesz *@
@{ } @* kód blokk *@
<!DOCTYPE html>

<html> <head>…</head> @* statikus tartalom *@
    <body>
        <div>
            @Model @* elhelyezzük a nézetmodellt az
                      oldalban *@
        </div>
    </body>
</html>
```

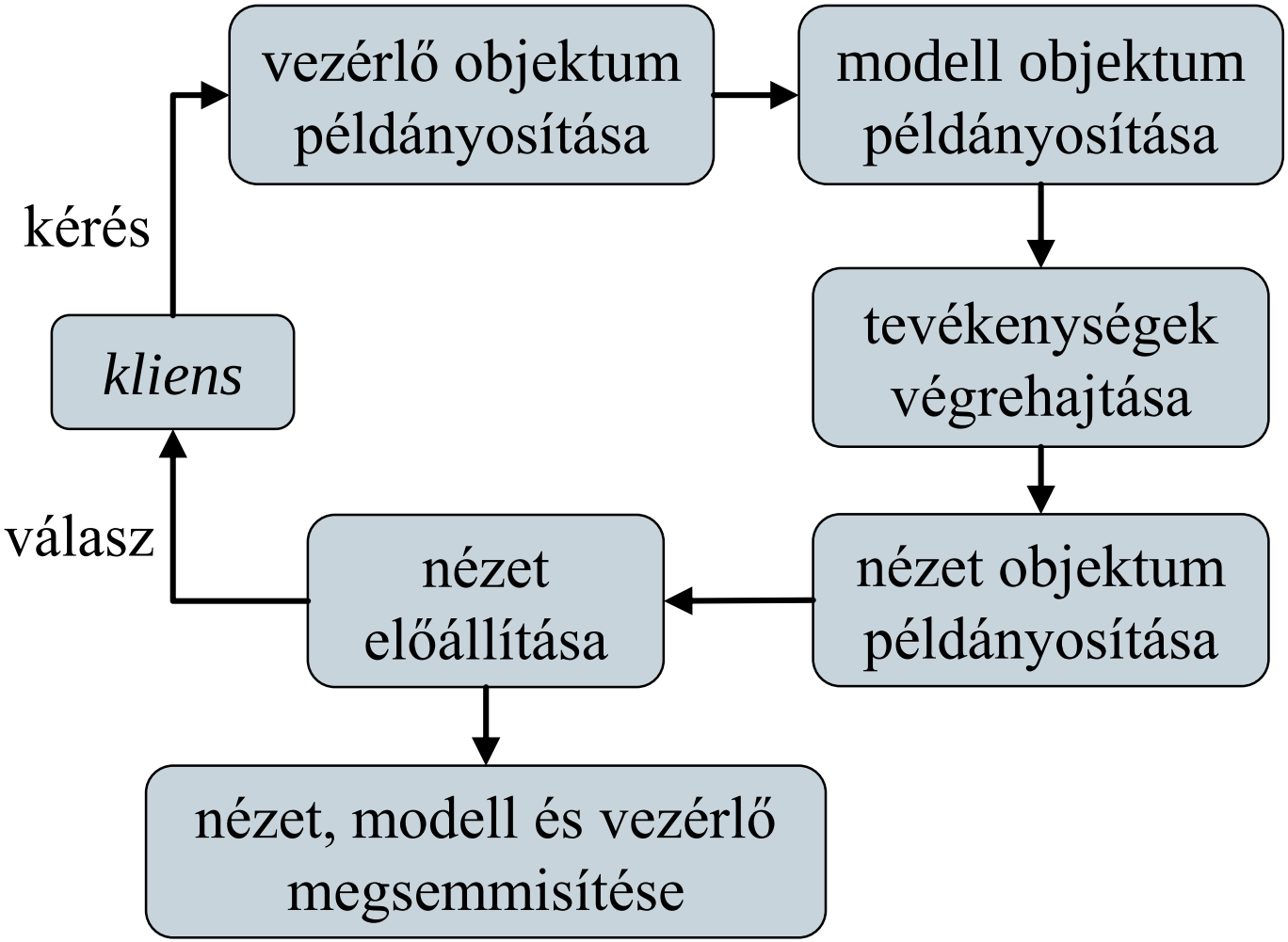
Webfejlesztés MVC architektúrában

Alkalmazás életciklus

- A webes alkalmazások életciklusa eltér az asztali és mobil alkalmazásokétól
  - az alkalmazás csak a kérésekre tud reagálni, a kérések függetlenek egymástól, és tetszőleges időpontban ér kezdhetnek
  - *az alkalmazás ezért két kérés között nem őrzi meg az állapotot*
    - kérés hatására indul, és példányosítja az objektumokat
    - a kérés kiszolgálásával törli az objektumokat
    - bizonyos adatok egy ideig a memóriában maradnak
  - *a perzisztencia réteg biztosítja az adatok megőrzését*

Webfejlesztés MVC architektúrában

Alkalmazás életciklus



Webfejlesztés MVC architektúrában

Weblapok hierarchiája

- Az MVC alkalmazás könyvtárfelépítése tükrözi a moduláris felépítést
  - a **`View`**, **`Controllers`** és **`Models`** könyvtárak a megfelelő tartalmat hordozzák
  - az **`App_Data`** könyvtár tárolja az esetleges adattartalmat (pl. adatbázis fájlok)
  - az **`App_Start`** könyvtár az indítási tevékenységeket (pl. **`RouteConfig`**)
  - a gyökérben található a konfiguráció (**`web.config`**), valamint az alkalmazásszintű vezérlés (**`Global.asax`**)
- MVC alkalmazáshoz szükséges a *NuGet* csomagkezelő

Webfejlesztés MVC architektúrában

Elérési útvonalak kezelése

- Az MVC architektúrában a felhasználó a vezérlővel létesít kapcsolatot, és annak akcióit futtatja (paraméterekkel)
  - az elérés és paraméterezés útvonalak segítségével adott, amelyek egy útvonalkezelő (*routing engine*) felügyel
  - az elérés testre szabható (`RouteConfig`), alapértelmezetten a `<domain>/<vezérlő>/<akció>/<paraméterek>` formában biztosított
    - vezérlő megadása nélkül az alapértelmezett `HomeController` vezérlőt tölti be
    - akció megadása nélkül az `Index` akciót futtatja
    - a paraméterek feloldása sorrendben, vagy név alapján történhet

Webfejlesztés MVC architektúrában

Elérési útvonalak kezelése

- pl.

Útvonal	*Tevékenység*
`http://myPage/`	a **`HomeController`** vezérlő **`Index`** metódusa fut
`http://myPage/Hello`	a **`HelloController`** vezérlő **`Index`** művelete fut
`…/Hello/List`	a **`HelloController`** vezérlő **`List`** művelete fut
`…/Hello/Details/1` **`…/Hello/Details?id=1`**	a **`HelloController`** vezérlő **`Details`** művelete fut **`id=1`** paraméterrel

Webfejlesztés MVC architektúrában

Vezérlők

- A vezérlők a **`Controller`** osztály leszármazottai, amelyek az akciókat publikus műveletek segítségével valósítják meg
  - a tevékenység egy eredményt ad vissza (**`ActionResult`**), amely lehet
    - nézet (**`ViewResult`**, **`PartialViewResult`**)
    - hibajelzés (**`HttpNotFoundResult`**, **`HttpUnathorizedResult`**, **`HttpStatusCodeResult`**)
    - átirányítás (**`RedirectResult`**)
    - fájl (**`FileResult`**), szkript (**`JavaScriptResult`**), egyéb tartalom (**`ContentResult`**)
    - üres (**`EmptyResult`**)

Webfejlesztés MVC architektúrában

Vezérlők

- pl.:

```
public class MyController : Controller {
    …
    public ActionResult LoadImage(String id){
        Byte[] image = … // adat betöltése
        if (image == null) // rossz az azonosító
            return RedirectToAction("NotFound");
            // átirányítunk egy másik akcióhoz
        return File(image, "image/png");
            // visszaadjuk fájlként a tartalmat
    }
    public HttpNotFoundResult NotFound() {
        return HttpNotFound("Content not found.");
    } // hibajelzés
```

Webfejlesztés MVC architektúrában

Vezérlők

- az eredménytípusokhoz tartozik egy művelet a `Controller` osztályban, amely azt előállítja, pl.:

```
return View(…); // eredmény ViewResult lesz
```

- a nézethez általában megadjuk a nézetmodellt, amely a modell leszűkítése és transzformációja a nézetre
  - pl.:

```
Object viewModel = …
    // létrehozzuk a nézetmodellt
return View("Index", viewModel);
    // megadjuk a nézet nevét és a
    // nézetmodellt
```

  - a nézetmodell tetszőleges típusú lehet, akár primitív is, és lehet teljesen független az eredeti modelltől

Webfejlesztés MVC architektúrában

Nézetek

- A nézet több leíró nyelvet is támogat, ezek közül a Razor rendelkezik a legegyszerűbb szintaxissal
  - a nézet lehet erősen típusos, ekkor megadjuk a nézetmodell típusát, pl. `@model MyProject.Model.ItemModel`
  - a dinamikus elemeket a `@` előtaggal jelöljük
    - a `@{ … }` blokkban tetszőleges háttérkódot helyezhetünk
    - a `@* … *@` blokk jelöli a kommentet
    - használhatunk elágazásokat (`@if`) és ciklusokat (`@for`, `@foreach`)
    - megadhatunk névtérhasználatot a `@using` elemmel (ellenkezdő esetben a teljes elérési útvonalat használjuk)

Webfejlesztés MVC architektúrában

Nézetek

- a nézetmodellre a **`Model`** elemmel hivatkozhatunk
- speciális HTML tartalmat a **`Html`** elemmel érhetünk el, pl.:
  - hivatkozások akciókra (**`ActionLink`**), amelyben megadjuk az akciót, (a vezérlőt) és az argumentumokat (egy anonim objektumban), pl.:

```
Html.ActionLink("LoadImage", new { id = 1 });
    // fordítás után LoadImage?id=1 link lesz
```

  - űrlapok (**`BeginForm`**, **`EndForm`**)
  - megjelenítő és beolvasó elemek (**`LabelFor`**, **`TextBoxFor`**, **`PasswordFor`**), ellenőrzések (**`ValidationMessageFor`**) űrlapok számára
  - nem kódolt tartalom elhelyezése (**`Raw`**)

Webfejlesztés MVC architektúrában

Nézetek

- lekérdezhetjük a kérés típusát (`IsPost`)
- speciálisabb elérési útvonalakat az `Url` elemmel kezelhetjük, pl.: `Url.Content("~/style.css")`
- beágyazhatunk szkripteket a `Scripts` elemmel, stílusokat a `Styles` elemmel, pl.:
  `Scripts.Render("~/bundles/jqueryval")`
- a nézetnek megadhatunk elrendezéseket (`Layout`), illetve különböző profilokhoz igazíthatjuk őket (pl. asztali/mobil környezet)

- A nézet egy olyan objektum, amely megvalósítja az `IView` interfészt, a nézet leíró nyelve (motorja) pedig az `IViewEngine` interfészt, így lehet saját motorokat és nézeteket megvalósítani

Webfejlesztés MVC architektúrában

Nézetek

- Amennyiben nem szeretnénk külön nézetmodellt használni, lehetőségünk külön a nézet számára információkat és akár tevékenységeket is átadni a **`ViewBag`** tulajdonságon keresztül a vezérlőben
  - egy dinamikusan kezelt, **`ExpandoObject`** objektum, azaz tetszőleges tulajdonsággal, illetve metódussal ruházható fel így bármilyen értéket tud fogadni, pl.:

```
ViewBag.Message = "Hello";
    // létrehozzuk a Message tulajdonságot, és
    // értéket adunk neki
```

  - a tartalmat a nézet elérheti és felhasználhatja, pl.:

```
<div>@ViewBag.Message</div>
```

Webfejlesztés MVC architektúrában

Nézetek

- pl.:

```
public class MyController : Controller {
    …
    public ActionResult List(){
        ViewBag.Title = "List of names";
            // beállítjuk az oldal címét
        String[] itemNames = …;
        return View("ListPage", itemNames);
            // visszaadunk egy tömböt a List
            // nézetnek
    }
    public ActionResult Details(String name){
        return View("DetailsPage", …);
    }
```

Webfejlesztés MVC architektúrában

Nézetek

- pl. (`ListPage.cshtml`):

```
@model IEnumerable<String>
@* erősen típusos oldal *@
@{
    Layout = null; // nincs elrendezés
}
<!DOCTYPE html>

<html>
    <head>
        <title>@ViewBag.Title</title>
    </head> @* a címet a vezérlő adja meg *@
    <body>
```

Webfejlesztés MVC architektúrában

Nézetek

```
        @if (Model != null){ // elágazás
            <div>
                @foreach (String name in Model){
                    <span><b>@name</b>
                    @Html.ActionLink("See details",
                            "Details", name);
                    @* link a Details akcióra *@
                    </span>
                }
            </div>
        } else {
            <div>No items found!</div>
        }
    </body></html>
```

Webfejlesztés MVC architektúrában

Weblapok kihelyezése

- A weblapokat a fejlesztést követően ki kell helyezni (*deploy*) egy webszerverre
  - a konfigurációnak (**`web.config`**) megadható egy nyomkövetési (*debug*) és egy kiadási (*release*) változata
  - a weblap alapból nyomkövetésre van konfigurálva, ezt érdemes kikapcsolni a **`compilation`** elemben
  - biztonsági okokból a kivételek jelzését csak lokálisan engedélyezzük a **`customErrors`** beállításával
  - a nézetek csak a futtatás során fordulnak le, ezért külön oda kell figyelni a hibaellenőrzésre (ez felüldefiniálható a projektfájlban)