Webes alkalmazások fejlesztése - Bevezetés

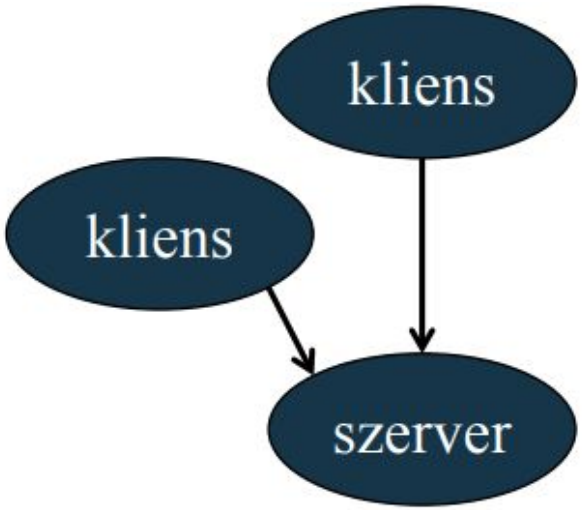
Jánosi-Rancz Katalin Tünde

Sapientia EMTE
tsuto@ms.sapientia.ro

Rendszerek

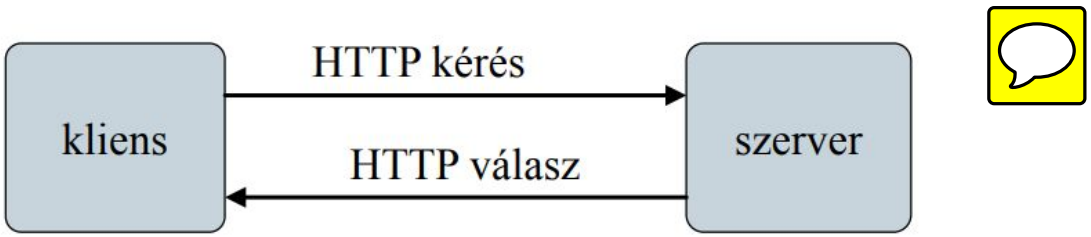
- A hálózaton kommunikáló alkalmazás-rendszerek alapvetően két kategóriába sorolhatóak:
  - decentralizált, közvetlen kapcsolatú (P2P)
  - `kliens-szerver` : egy központ gép látja el a funkciókat, és szolgál ki tetszőleges sok csatlakozó gépet, amely lehet:

- vastagkliens (thick client): a végrehajtást a kliens végzi, a szerver főleg kapcsolattartásra, adatkezelésre szolgál
- vékonykliens (thin client): a végrehajtást a szerver végzi, a kliens interakcióra szolgál



A HTTP protokoll

- A webes adatközlés alapvető protokollja a `HTTP` (Hypertext Transfer Protocol)
  - a kérés/válasz paradigmára épül, vagyis a kliens elküld egy kérést, amelyre a szerver válaszol
    - a választ értelmezi és megjeleníti a kliens
    - a válasz első sora a státuszsor, amely visszajelez a kérés végrehajtásáról (pl. 200 sikeres, 404 nem található, 503 szolgáltatás nem elérhető)
- a protokoll biztonságossá tehető TSL titkosítással (HTTPS)



A webes alkalmazások életciklusa

- az alkalmazás csak a kérésekre tud reagálni, a kérések függetlenek egymástól, és tetszőleges időpontban érkezhetnek
  - a kérés lehet pl. `GET` (adott erőforrás lekérése), `POST` (adatokkal ellátott tartalom küldése)
  - ASP.NET eldönti, hogy szükség van-e az oldal feldolgozására vagy visszaküldhető a cache-ben tárolt változat
- az alkalmazás két kérés között nem őrzi meg az állapotot
  - kérés hatására indul, és példányosítja az objektumokat
  - a szerveren történt feldolgozást követően a generált tartalom visszamegy és meg lesz jelenítve (kimenetgenerálás - `rendering`) a böngészőben
  - a kérés kiszolgálásával törli az objektumokat
  - bizonyos adatok egy ideig a memóriában maradnak
- ugyanakkor a szerver felépíthet egy munkamenetet (`session`) a kliens felé (az azonosító adatok kliens oldalon kerülnek mentésre)
- a perzisztenciaréteg biztosítja az adatok megőrzését

ASP.NET

Három rétegű (N-rétegű) alkalmazások

Presentation tier

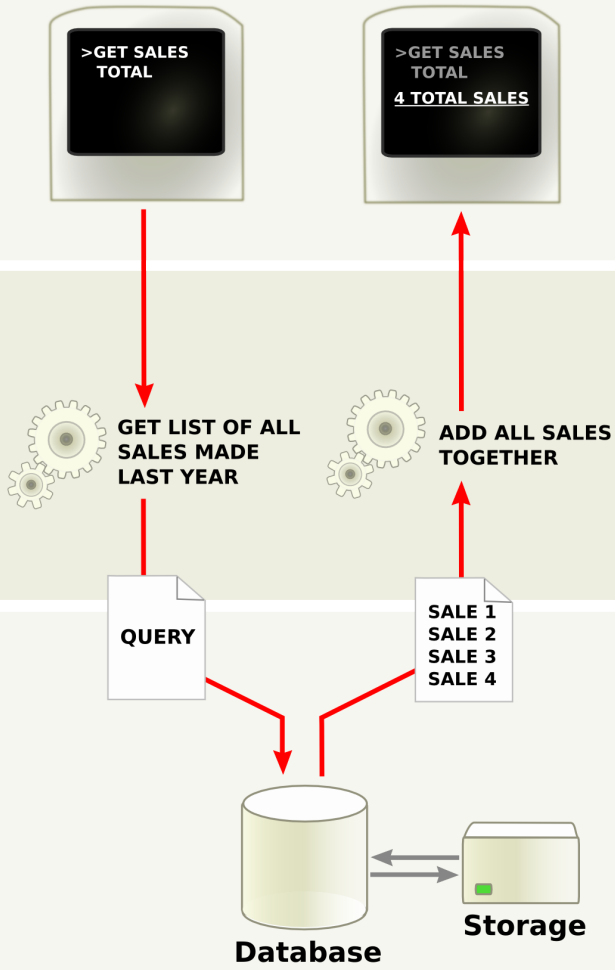
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

Data tier

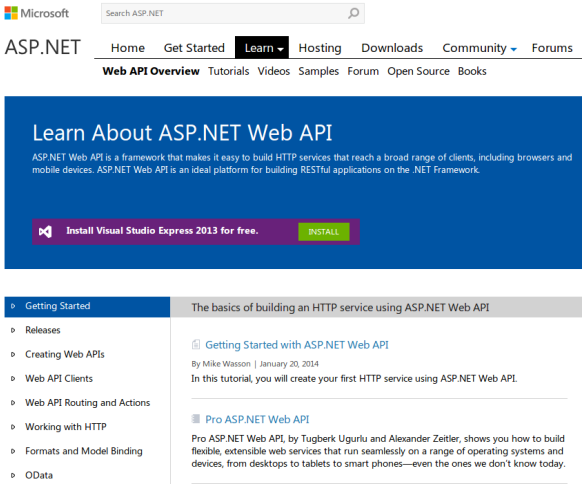
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

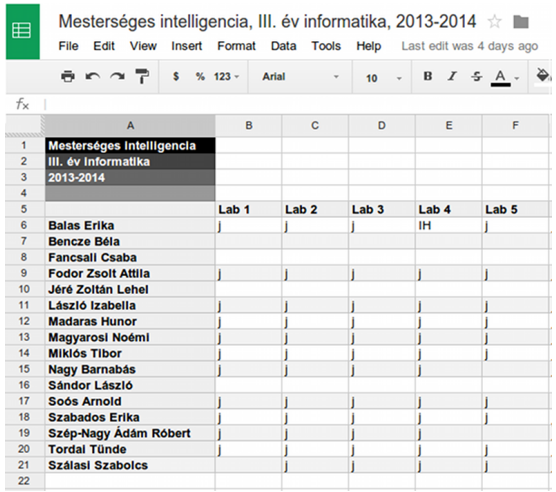


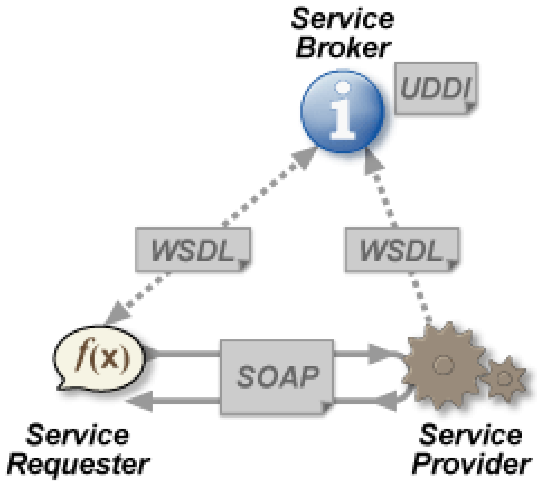
Source: Wikipedia

ASP.NET

- az ASP (Active Server Pages) továbbfejlesztése .NET programozás támogatással
- egy (szerver oldali) programozási felület dinamikus weboldalak, webes alkalmazások, webszolgáltatások készítésére HTTP protokollon keresztül
- IIS (Internet Information Services) szerveren fut





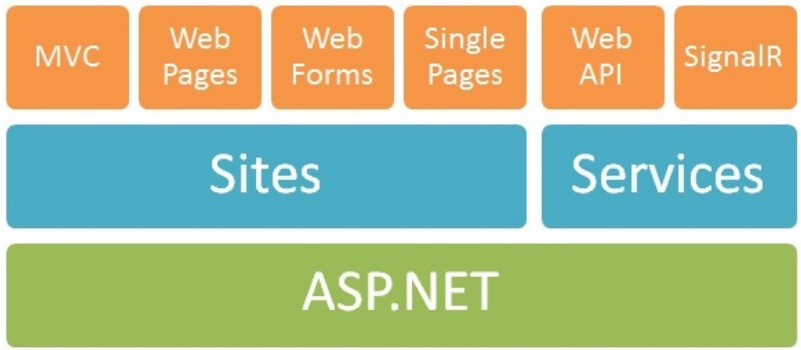


Miért ASP.NET?

- több programozási nyelvet támogat
- programozható kontrolok
- eseményvezérelt programozás
- XML alapú komponensek
- felhasználó azonosítás (felhasználói fiokok, szerepek)
- jobb skálázhatóság
- hatékonyság (lefordított kód)
- egyszerűbb konfigurálás és telepítés
- dinamikusan méretezhető, több device-on nézhető

ASP.NET

- 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
- struktúrája:



Webfejlesztés MVC architektúrában

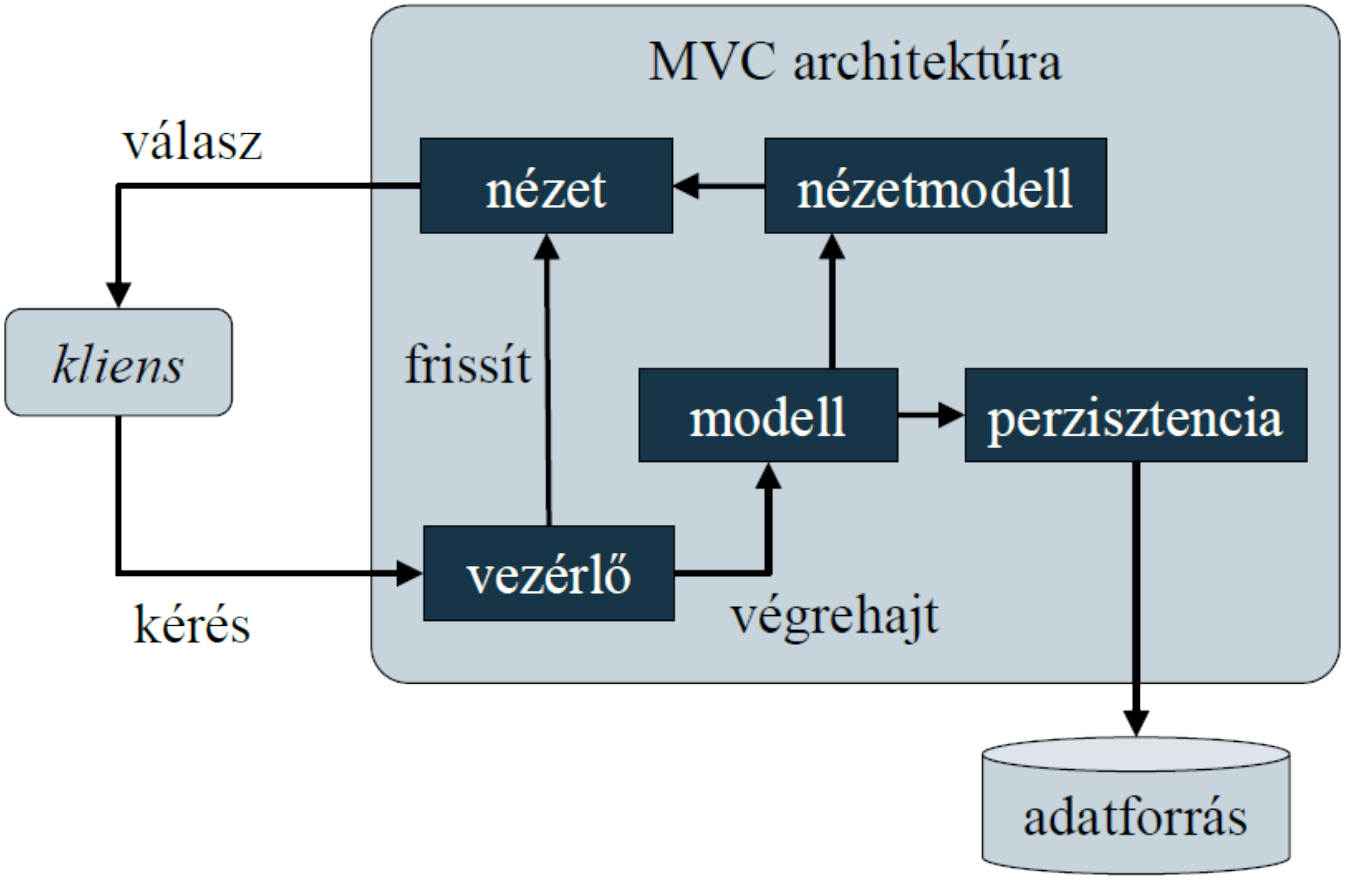
Webfejlesztés MVC architektúrában

- desktop alkalmazásoknál 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 Controller az alkalmazás belépési pontja
  - a vezérlésre az alkalmazásnak egy (új) nézettel kell válaszolnia, ami az adott erőforráshoz tartozik

A modell/nézet/vezérlő (Model-View-Controller, MVC)

- a `vezérlő` a kérések kiszolgálója, amely biztosítja a nézetet a kérés eredménye alapján
- a `nézet` csak a megjelenítéssel foglalkozik, nem tartalmaz háttérkódot, csupán az adatokat kéri a modelltől
  - a `nézetben` adatkötéssel hivatkozhatunk a `modell` tartalmára, illetve használhatunk HTML kódot
- a `modell` a logikai funkciók végrehajtása
- a `nézetmodell` egy átjáró, amely az adatokat a `nézet` számára megfelelő módon prezentálja
- a perzisztencia felel az adatelérésért (tetszőleges)

MVC architektúra



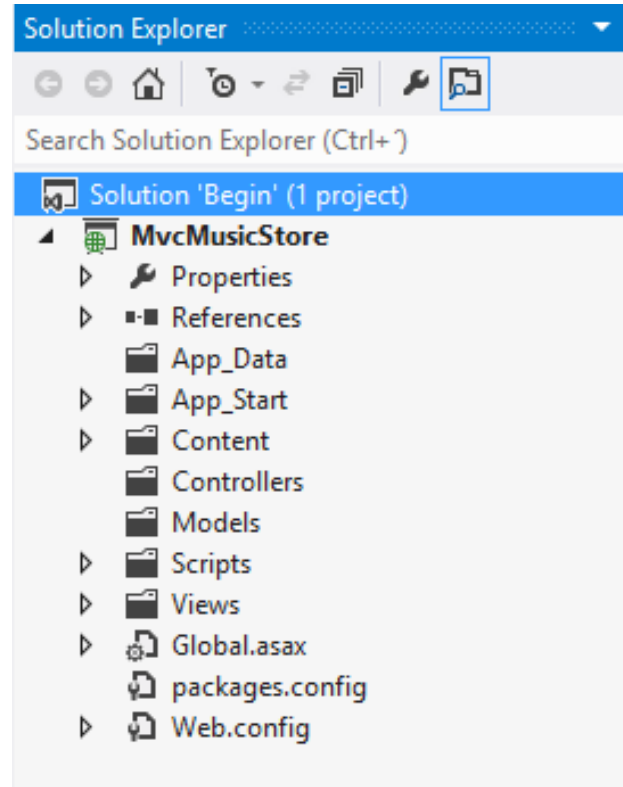
Source: Webes alkalmazások fejlesztése, Giachetta Roberto, http://people.inf.elte.hu/groberto

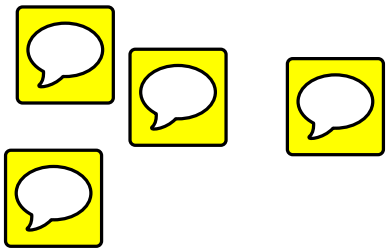
Az MVC architektúra végrehajtási ciklusa

- a felhasználó egy kérést ad a szervernek
- a vezérlő fogadja a kérést, majd a modellben végrehajtja a megfelelő akciót (`action method`)
- a modellben végrehajtott akció állapotváltást okoz
- 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
- a felhasználó megkapja a választ (nézetet)

Weblapok hierarchiája

- a Views, 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`)





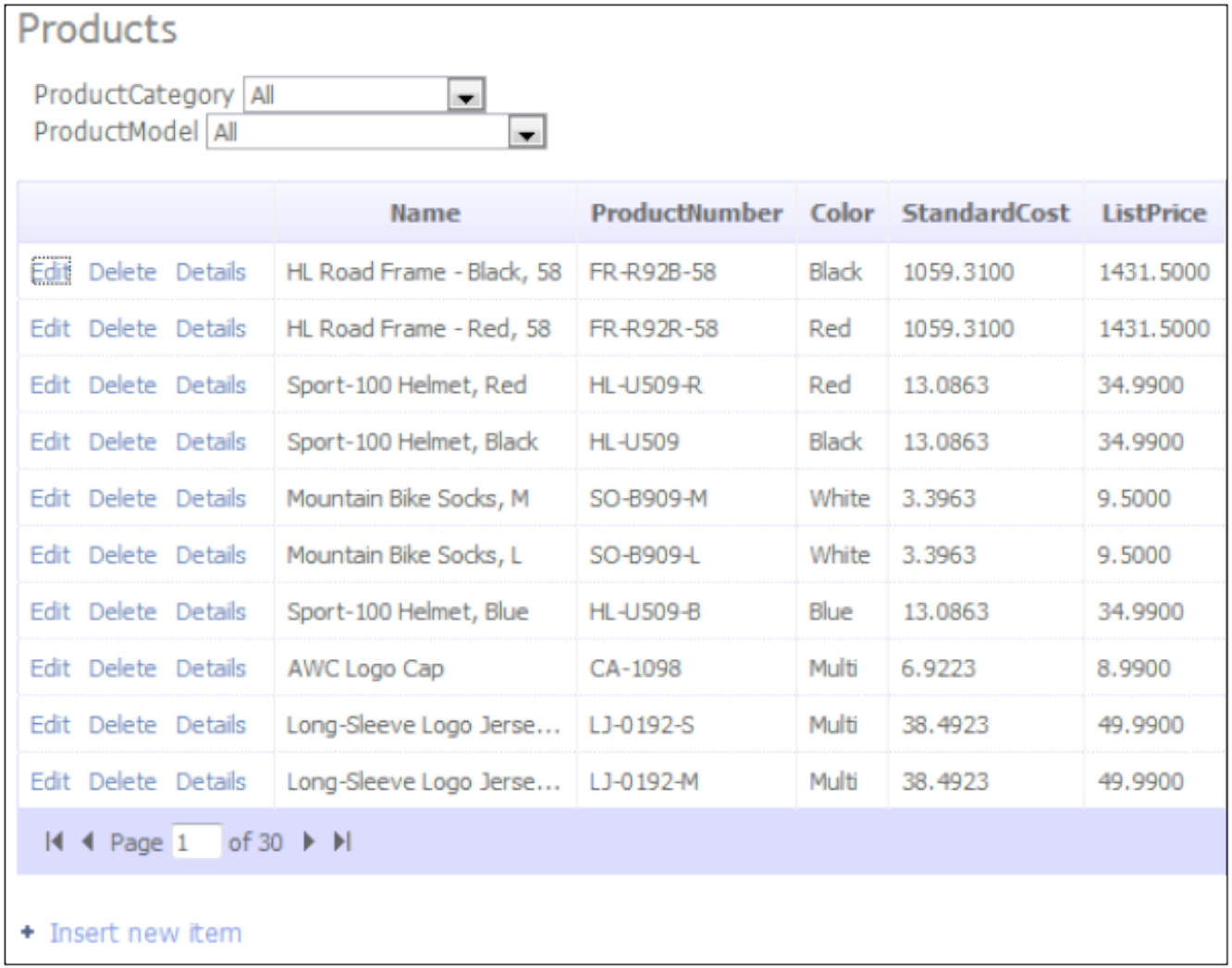
Objektumrelációs adatkezelés

- Entity Framework

Dinamikus adatok ASP.NET-ben

- segítségével adatvezérelt alkalmazásokat készíthetünk Linq vagy EF modell alapján
- testreszabható
- adatbázisokat karbantartó ablakok létrehozását teszi lehetővé
- dinamikus adat elemeket generál az adatbázis összes táblájára
- CRUD műveleteket biztosít
- automatikus validálás
- adatszűrés
- támogatja az N-N kapcsolatot

Dinamikus adatok ASP.NET-ben, példa



Source: Dan Wahlin's course. (Pluralsight)

Weblapok kihelyezése

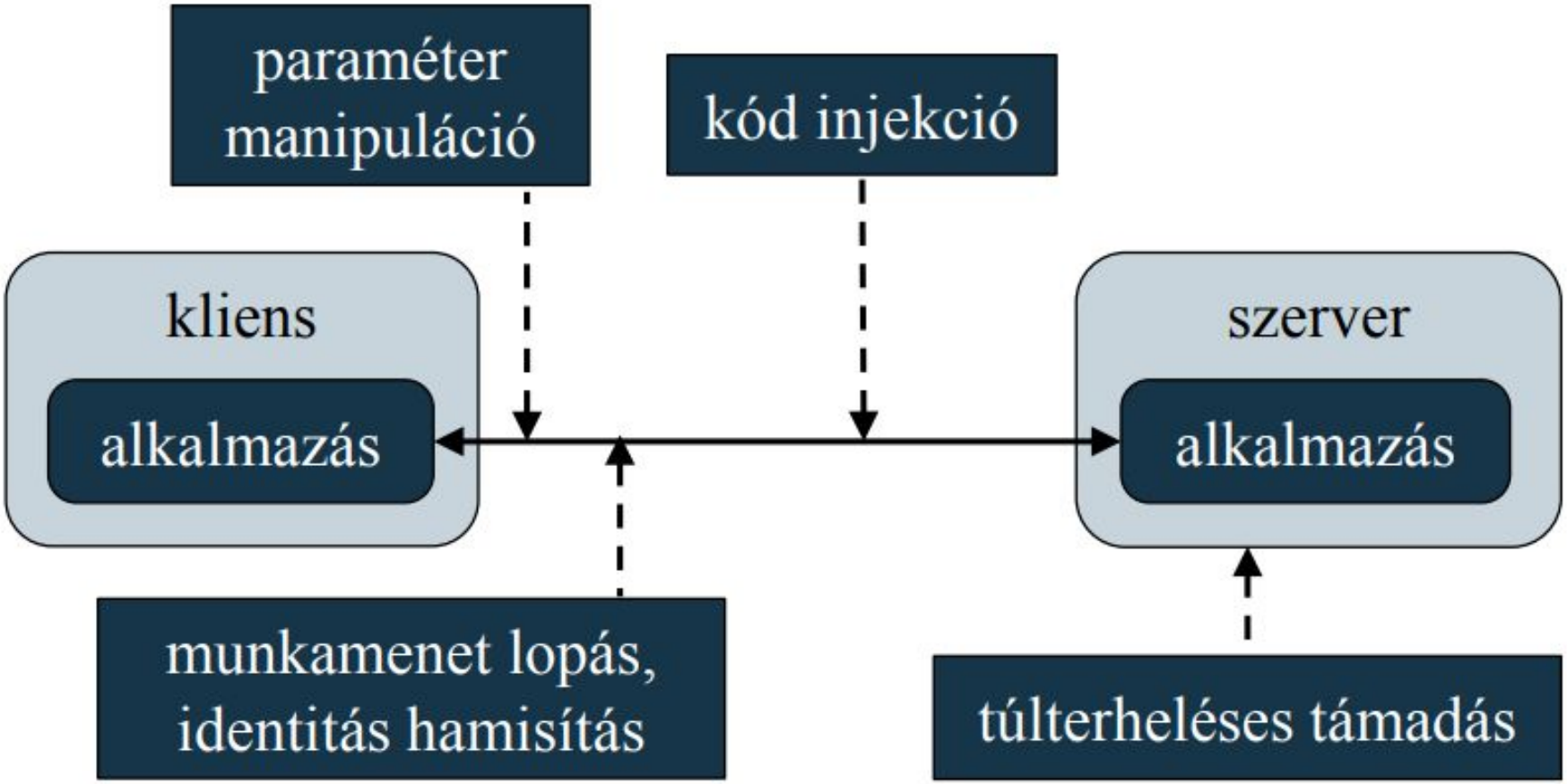
- a weblapokat a fejlesztést követően ki kell helyezni (deploy) egy webszerverre
- a konfigurációnak (web.config) megadható egy nyomkövetési (debug) és egy kiadási (release) változata
- a weblap alapból nyomkövetésre van konfigurálva, ezt érdemes kikapcsolni a compilation elemben
- biztonsági okokból 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)

Tesztelhetőség

- az egységtesztelés (Unit Test) segít a kód magabiztos megváltoztatásában
- az ASP.NET MVC-t úgy tervezték, hogy egyszerűbbé tegye az egységtesztelést

Webes alkalmazások és biztonságuk

Külön hangsúlyt kell helyezni a biztonságra, mivel számos ponton támadás érheti.



Biztonság

- a kulcsfontosságú adatokat lehetőség szerint megfelelően titkosítani kell
- SQL injekció: a szerveren futó SQL utasításokat manipulálja

```
textPass.Text = ”’ or ’1’ = ’1”
```

- cross-site scripting (XSS): szkript kerül feltöltésre a szerverre, amelyet a kliens böngészője futtat

```
labelPost.Text = textInput.Text;
// textInput.Text = ”<script…></script>” esetén lefut a script a
     válaszban
```

- gyenge kivételkezelés, és a kivétel információk megjelenése kliens oldalon (számos belső információt jeleníthet meg, amely tovább könnyíti a behatolást)
- oldal argumentumokat ne ruházzunk fel túl nagy felelősséggel, pl. elérési útvonalak megadása, felhasználói azonosítás (munkamenetek helyett)

```
order.php?id=245601&disc=0    // disc=100 esetén ingyenes a rendelés
```

Biztonság 2

- felhasználói információkat ne tároljuk kliens oldalon, pl. felhasználónév és jelszó sütiben
- a munkamenetet általában sütik segítségével tárolják a kliens oldalon, ezért a süti megszerzése egyben a munkafolyamathoz való hozzáférést is biztosítja (munkamenet lopás (session hijacking))
  - sütiket biztonságos csatornán továbbítsuk
  - a munkamenet lejáratát korlátozzuk
  - a munkamenetet sütin kívül is authentikáljuk
- kritikus információ tárolására ne használjunk rejtett mezőket

Forrás

Giachetta Roberto: Webes alkalmazások fejlesztése alapján, http://people.inf.elte.hu/groberto