



**Eötvös Loránd Tudományegyetem
Informatikai Kar**

Webes alkalmazások fejlesztése

6. fejezet

Állapotfenntartás (ASP.NET)

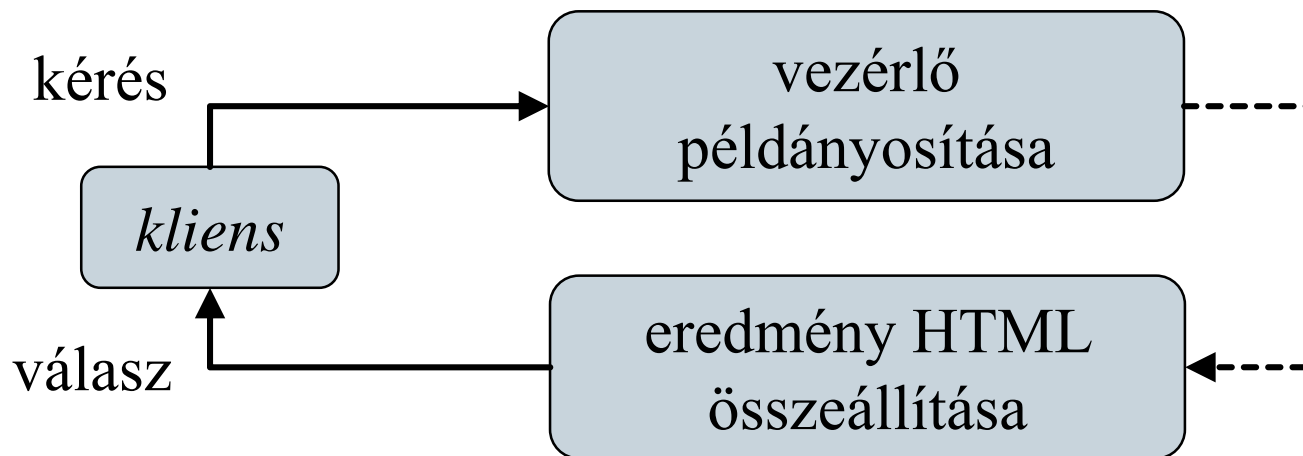
Giachetta Roberto

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

Állapotfenntartás

A HTTP protokoll

- A HTTP protokoll a kérés/válasz paradigmára épül, vagyis a kliens elküld egy kérést, amelyre a szerver (alkalmazás) válaszol
 - a kérések egymástól függetlenül kerülnek kiszolgálásra
 - minden kiszolgáláshoz külön objektumok jönnek létre, amelyek előállítják a választ, majd megsemmisülnek



Állapotfenntartás

Eszközök

- Két kérés között sokszor szeretnénk megőrizni az állapotot
 - pl. a felhasználó bejelentkeztetése, az űrlap mezők kitöltései
 - objektum erre nincs lehetőség (a mezők megsemmisülnek), osztályszinten pedig nincs garancia a megőrzésre
- Az állapotot a szerver speciális eszközökkel tudja fenntartani
 - kliens oldalon: elérési útvonal, weblap értékei (űrlapmezők, rejtett mezők), *sütik*
 - szerver oldalon:
 - egy kliensre: *munkamenet*
 - minden kliensre: *alkalmazás*

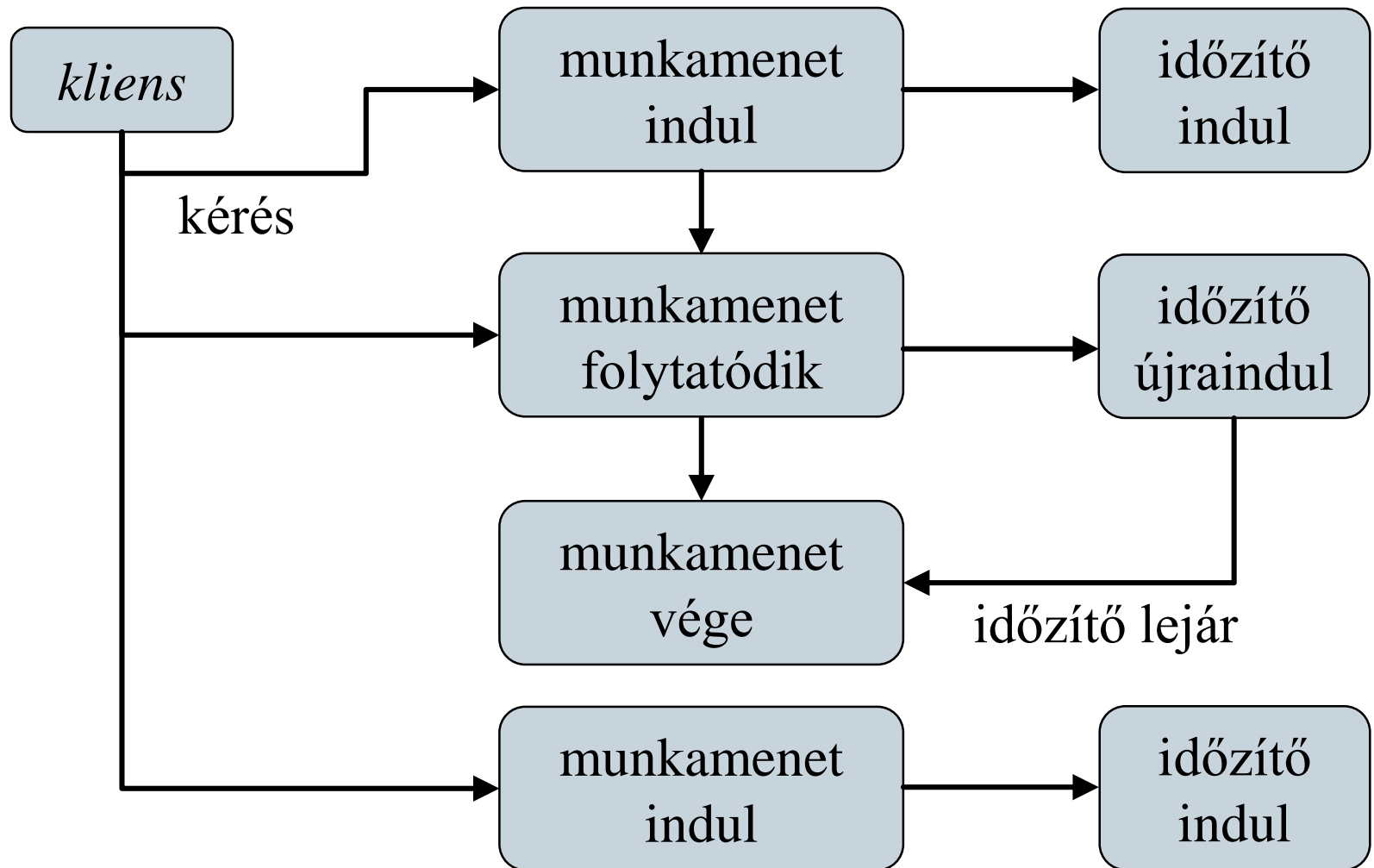
Állapotfenntartás

Munkamenet állapotok

- A munkamenet (*session*) egy kliens weblapon történő tartózkodása, és közben végrehajtott tevékenységei
 - minden kliens rendelkezik (pontosan) egy saját munkafolyamattal a szerver
 - automatikusan elindul, amikor a kliens először kérést küld a szerverre
 - automatikusan végződik, amikor a kliens egy megadott ideig nem intéz kérést, ezt egy időzítő felügyeli, amely minden kéréssel újraindul (*session timeout*)
 - a klienst a kérés paraméterei (IP cím, böngésző, ...) alapján azonosítja, ami meghamisítható (*session hijacking*)

Állapotfenntartás

Munkamenet állapotok



Állapotfenntartás

Munkamenet állapotok

- A munkamenethez szerver oldalon bármikor hozzáférhetünk a vezérlő/nézet **Session** tulajdonságán keresztül, vagy máshol a **HttpContext.Current.Session** tulajdonságon keresztül
 - ebben kulcs/érték párokként elhelyeztünk az adott kliensre vonatkozó adatokat, amelyeket a szerver a memóriában tárol (a munkafolyamat megszűnéséig), pl.:
Session["myKey"] = myValue;
 - a munkamenet azonosítója a **Session.SessionID** tulajdonsággal kérhető le
 - a munkamenet várakozási ideje a **Session.Timeout** tulajdonsággal állítható (alapértelmezetten 20 perc)
 - az adatokhoz a kliens nem férhet hozzá

- Az *adatlopás* elkerülése végett fontos, azonosításra szolgáló adatokat mindig kódoltan tároljuk az adatbázisban
 - a kódoláshoz egyirányú kódoló algoritmusokat használunk (pl. *MD5*, *SHA1*, *SHA512*), amelyek nem fejthetők vissza, viszont azonosítására használhatóak
 - a kódoló eljárások a **System.Security.Cryptography** névtérben helyezkednek el
 - a kódolás előtt és/vagy után célszerű megsózni a jelszót (*password salt*), azaz tegyünk bele extra karaktereket és byte-okat, hogy megnehezítsük a jelszó visszakeresését
 - a só lehet fix, véletlenszerű, vagy időfüggő, ilyenkor magát a sót is eltárolhatjuk az adatbázisban

Állapotfenntartás

Adattitkosítás

- Pl.:

```
String pwdText = ... // jelszó szöveges alakja
SHA512CryptoServiceProvider coder = ...
    // SHA512 kódoló objektum
```

```
Byte[] pwdBytes = coder.ComputeHash(
    Encoding.UTF8.GetBytes(pwdText));
    // kódolás végrehajtása a szövegből kiolvasott
    // UTF8 értékeken, az eredmény 160 bites lesz
```

```
Byte[] storedBytes = ...
    // kinyerjük az eltárolt kódolt jelszót
if (pwdBytes.SequenceEquals(storedBytes)) { ... }
    // ha a kettő megegyezik, jó a jelszó
```


Feladat: Valósítsuk az utazási ügynökség weblapjának felhasználó kezelési funkcióját.

- a felhasználók regisztrálhatnak, és adataikat foglaláskor automatikusan kitölti a weblap
 - a regisztráció nem kötelező, az újonnan megadott adatok a korábbiak szerint mentődnek (automatikusan generált felhasználónévvel)
- egy új vezérlőben (**AccountController**) kezeljük a regisztráció (**Register**), bejelentkezés (**Login**) és kijelentkezés (**Logout**) funkciókat
 - a regisztráció és a bejelentkezés megfelelő nézeteket kapnak, űrlapokkal

Állapotfenntartás

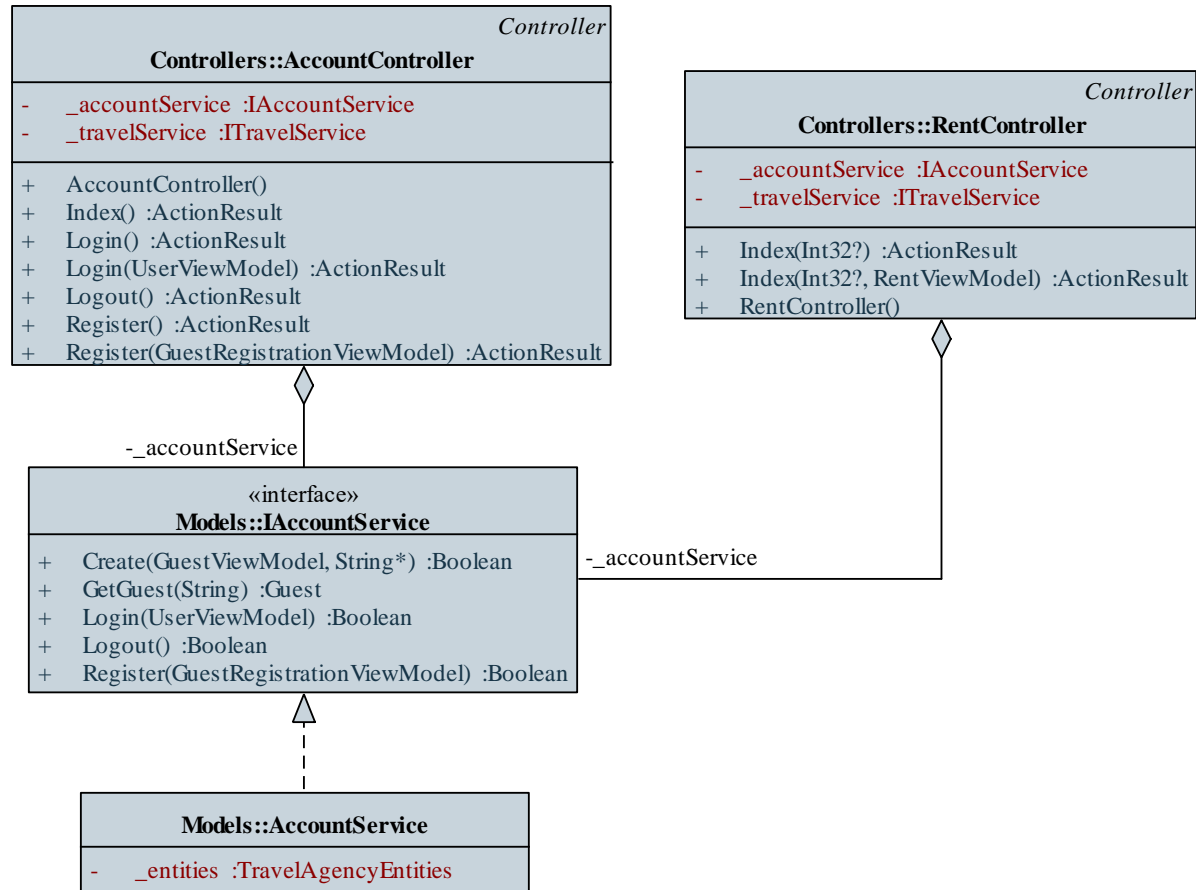
Példa

- a funkciókat az **AccountService** osztály hajtja végre, amely megvalósítja az **IAccountService** interfészt
 - a bejelentkezés, kijelentkezés és regisztráció mellett lekérhetjük egy adott vendég adatait (**GetGuest**), és létrehozhatunk vendéget regisztráció (felhasználói adatok) nélkül (**Create**)
- a nézetmodell bővül a bejelentkezés (**UserViewModel**), illetve a regisztráció (**GuestRegistrationViewModel**) adataival
 - mivel több adat közös a foglalás és a regisztráció között, egy ősosztályba (**GuestViewModel**) általánosítunk

Állapotfenntartás

Példa

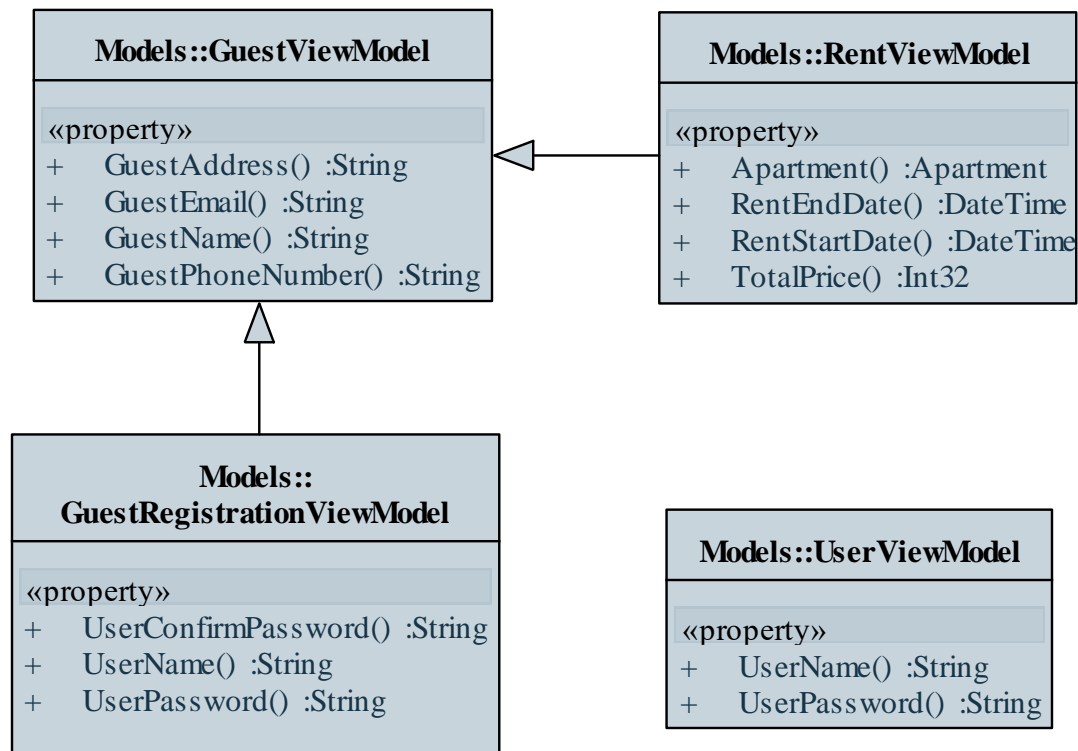
Tervezés (alkalmazás):



Állapotfenntartás

Példa

Tervezés (nézetmodellek):



Állapotfenntartás

Példa

Megvalósítás (`AccountController.cs`):

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Login(UserViewModel user) {
    ...
    Session["user"] = user.UserName;
    // felvesszük a felhasználó nevét a
    // munkamenetbe
    Session.Timeout = 15;
    // max. 15 percig él a munkamenet

    return RedirectToAction("Index", "Home");
    // átirányítjuk a főoldalra
}
```

Állapotfenntartás

Példa

Megvalósítás (`_Layout.cshtml`):

```
...
@if (Session["user"] == null) {
    // itt is hozzáférünk a munkafolyamathoz
    ...
} else {
    <table><tr>
        <td colspan="2"> Üdvözöljük,
                                @Session["user"]!</td>
    </tr><tr>
        <td>@Html.ActionLink("Kijelentkezés",
                                "Logout", "Account")</td>
    ...
} ...
```

Állapotfenntartás

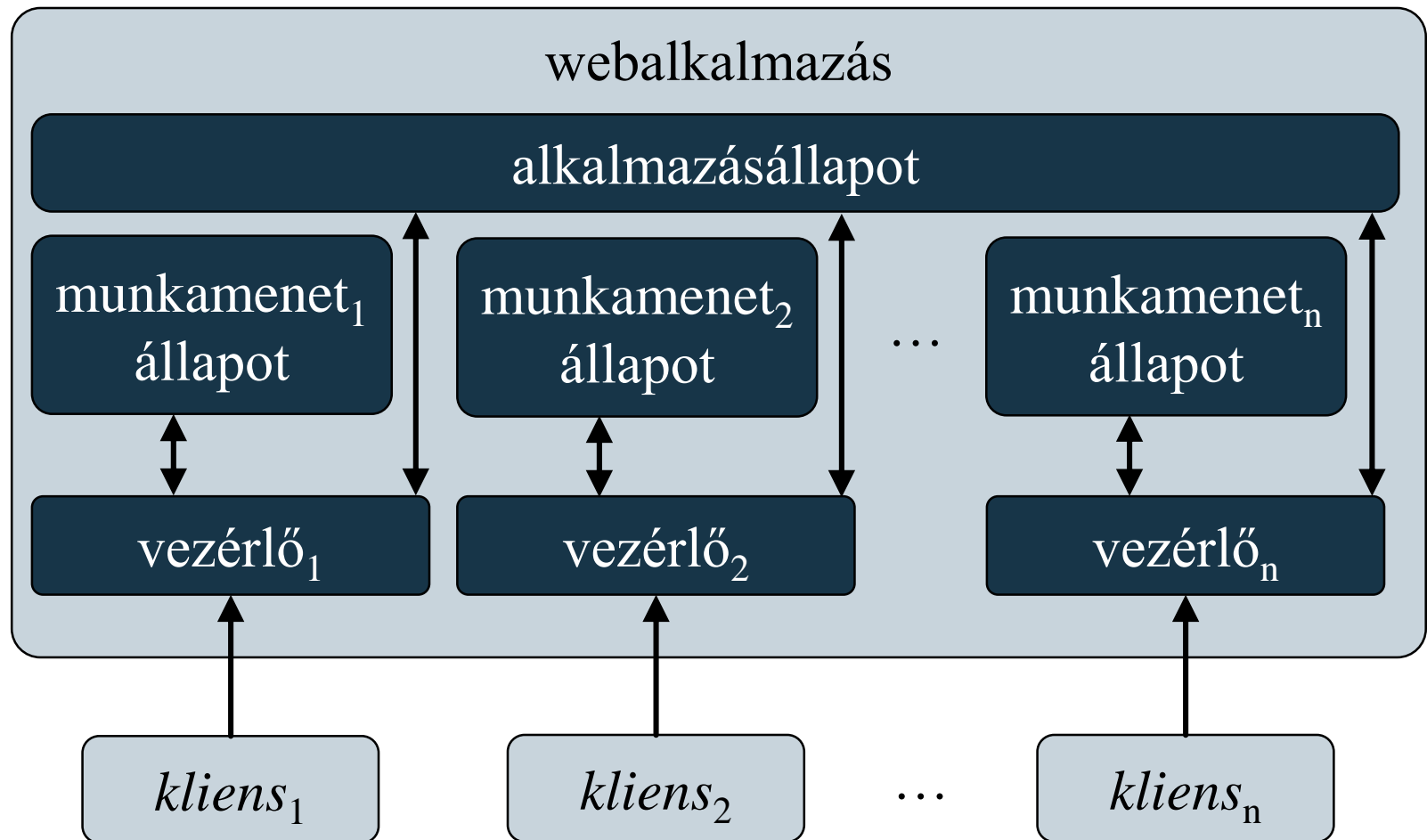
Alkalmazás állapotok

- Az egész web alkalmazásra vonatkozó, globális információkat is tárolhatunk a vezérlő `HttpContext.Application` vagy a nézet `HttpContext.Current.Application` tulajdonságán keresztül
 - pl.:

```
HttpContext.Application["myKey"] = myValue;  
// alkalmazás érték beállítása
```
 - minden vezérlőből ugyanahhoz a példányhoz férünk hozzá, így egyéni információk számára nem alkalmas
 - mivel párhuzamosan többen hozzáférhetnek, ezért célszerű kritikus szakaszba helyezni a beépített `Lock` és `Unlock` metódusokkal

Állapotfenntartás

Alkalmazás állapotok



Állapotfenntartás

Alkalmazás állapotok

- A web alkalmazásunk globális állapotkezeléséhez rendelkezésünkre áll az *alkalmazás osztály* (*Global Application Class*)
 - a **HttpApplication** leszármazottja, minden projektben egy található **Global.asax** néven
 - egy eseménykezelő halmazt tartalmaz, amelyek az alkalmazás, illetve az egyes munkamenetek kezdésére/végére, vagy hibajelenségek hatására futnak le
 - közvetlenül elérhetőek benne az alkalmazás (**Application**), illetve a munkamenetek (**Session**) értékei
 - alapból tartalmazza az útvonalak feloldásának konfigurációját

Állapotfenntartás

Alkalmazás állapotok

- pl.:
...

```
protected void Application_Start(...) {  
    ... // útvonalkonfiguráció betöltése  
    Application["sessionCount"] = 0;  
    // alkalmazásszintű változó  
}  
  
protected void Session_Start(...) {  
    Application["sessionCount"] =  
        (Int32) (Application["sessionCount"]) + 1;  
    // változtatás  
}  
...
```

- A HTTP *süti* (**HttpCookie**) olyan információgyűjtemény, amelyet a kliens eltárol egy fájlban, így az oldal későbbi látogatása során a felhasználóra vonatkozó adatok abból visszatölthetők
 - a süti adott webcímre vonatkoznak, és kulcs/érték párokat tartalmaznak (vagy csupán egy értéket), pl.:
`HttpCookie cookie = new HttpCookie("MyCookie");`
`cookie.Add("myKey", myValue);`
 - megadhatunk lejáratot, amely elteltével a süti törlődik, pl.:
`cookie.Expires = DateTime.Now.AddDays(10);`
 - az adott webcímhez tartozó sütiket a böngésző automatikusan továbbítja a kérésben

- Sütiket a vezérlőben kezelhetjük
 - a kéréssel küldött sütit a **Request.Cookies** gyűjteményben találjuk, pl.:
`HttpCookie c = Request.Cookies["MyCookie"];`
 - a válaszhoz sütit a **Response.Cookies** gyűjteménybe helyezhetjük, pl.:
`Response.Cookies.Add(cookie);`
 - sütit úgy törölhetünk, hogy az érvényességét lejárt időpontra állítjuk (és így a böngész kitörli)
- A munkafolyamatok klienseinek beazonosításához is sütit használunk, ez a munkafolyamat süti (**ASP.NET_SessionId**)

- Pl.:

```
[HttpPost]
public ActionResult LoginUser(UserData user) {
    ...
    if (user.RemberMe) {
        // ha kérte az azonosító megjegyzését
        HttpCookie c = new HttpCookie("uid");
        c["userName"] = user.UserName;
        Response.Cookies.Add(c);
        // az azonosítót elküldjük a kliensnek
    }
    return View(...);
}
```

Állapotfenntartás

Sütik

```
[HttpGet]
public ActionResult LoginUser() {
    UserData user = ...
    // amikor legközelebb betölti az oldalt

    if (Request.Cookies["uid"] != null) {
        // és megjegyeztette az azonosítót

        user.UserName = HttpContext.Request.
            Cookies["uid"]["userName"].ToString();
        // beállítjuk előre az azonosítót
    }
    return View(user);
}
```

Állapotfenntartás

Példa

Feladat: Valósítsuk az utazási ügynökség weblapjának felhasználó kezelési funkcióját.

- lehetőséget adunk a felhasználónak a bejelentkezés megjegyzésére
 - az azonosító megjegyzéséhez sütit használunk, amelyet bejelentkezést követően továbbítunk a felhasználónak (a sütiben a felhasználónevet tároljuk)
 - minden munkafolyamat indulásakor (**Session_Start**) ellenőrizzük a süti jelenlétét
- a felületen megjelenítjük az oldalt böngésző felhasználók számát, ezt alkalmazás állapotban tároljuk

Állapotfenntartás

Példa

Megvalósítás (AccountController.cs):

```
public void Login(UserViewModel user) {  
    ...  
    if (user.RememberLogin) {  
        // ha meg kell jegyeznünk a felhasználónevet  
        HttpCookie cookie = new HttpCookie("user");  
        // akkor elküldjük azt sütiként  
        cookie["userName"] = login.UserName;  
        cookie.Expires =  
            DateTime.Today.AddDays(365);  
        // egy évig lesz érvényes a süti  
        Response.Cookies.Add(cookie);  
    }  
    ...  
}
```


Állapotfenntartás

Példa

Megvalósítás (Global.asax.cs):

...

```
protected void Session_Start() {  
    // munkafolyamat indulása  
    if (Request.Cookies["user"] != null) {  
        Session["user"] =  
            Request.Cookies["user"]["userName"];  
        // felvesszük a felhasználó nevét a  
        // munkamenetbe  
        Session.Timeout = 15;  
        // max. 15 percig él a munkamenet  
    }  
}
```

Állapotfenntartás

Sütik biztonságos kezelése

- Mivel a sütik szolgáltatják a kliens oldali információ tárolás (és benne a munkamenet tárolás) alapját, különösen figyelni kell a biztonságukra
 - felhasználói adatokat (különösen jelszavakat) direkt módon ne tároljuk sütiben
 - a sütik tartalmát kódolhatjuk, vagy helyettesíthetjük speciális azonosítókkal
 - szabályozható, hogy kliens oldali szkriptek ne férjenek hozzá a sütihez (**HttpOnly**)
 - szabályozható, hogy csak biztonságos (TSL/SSL) kapcsolat esetén továbbítódjanak (**Secure**)

Állapotfenntartás

Sütik biztonságos kezelése

- Amennyiben sütiket használunk a felhasználó azonosítására, különös tekintettel kell lennünk a biztonságra
 - az információt osszuk el több sütibe
 - jelszavak helyett használjunk egyedi azonosítókat (**Guid**), amelyeket mindkét oldalon eltárolunk
 - az azonosító cserélhetjük minden bejelentkezéssel
 - a felhasználói azonosítók mellett tárolhatunk felhasználó-specifikus információkat
 - a **Request** tulajdonság számos információt tartalmaz a kliensről (**UserHostAddress**, **UserHostName**, ...), amik szintén elmenthetők (kódolva) a sütibe

Állapotfenntartás

Példa

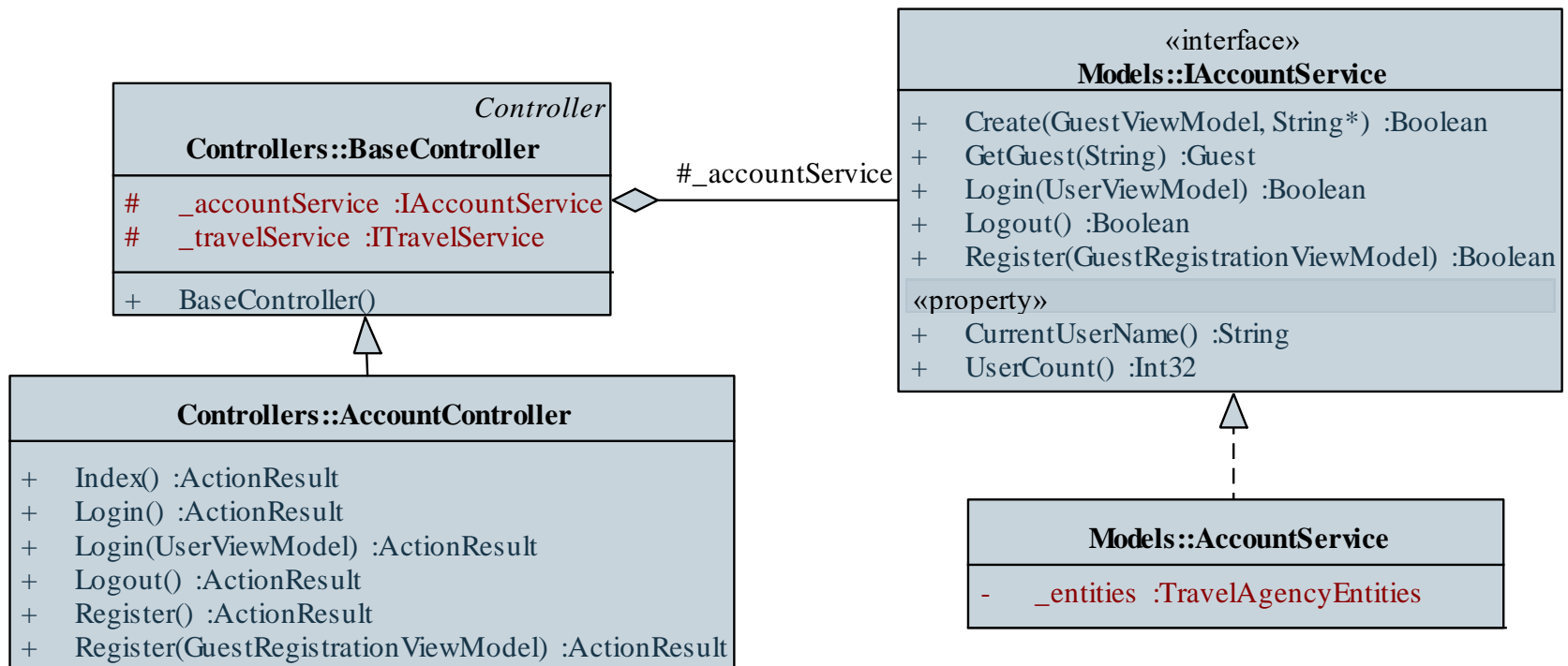
Feladat: Valósítsuk az utazási ügynökség weblapjának felhasználó kezelési funkcióját.

- a munkafolyamat és az alkalmazásállapot kezelését áthárítjuk az **AccountService** osztályra, így a vezérlő mentesül az állapotkezeléstől
 - a konstruktor ellenőrzi a sütit, és tölti be a munkafolyamatba (így nincs szükség a **Session_Start()** metódusra)
 - tulajdonságok segítségével kérdezzük le az aktuális felhasználót (**CurrentUserName**), illetve a felhasználók számát (**UserCount**)

Állapotfenntartás

Példa

Tervezés (alkalmazás):



Állapotfenntartás

Példa

Megvalósítás (AccountService.cs):

```
public AccountService() {  
    ...  
    if (HttpContext.Current.Request.Cookies["user"]  
        != null &&  
        HttpContext.Current.Session["user"] ==  
            null) {  
        HttpContext.Current.Session["user"] =  
            HttpContext.Current.Request.  
                Cookies["user"]["userName"];  
        // felvesszük a felhasználó nevét a  
        // munkamenetbe  
        ...  
    }  
}
```