Eötvös Loránd Tudományegyetem
Informatikai Kar

Webes alkalmazások fejlesztése

13. fejezet

Kommunikációs megoldások szolgáltatásokban (WCF)

Giachetta Roberto

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

Kommunikációs megoldások szolgáltatásokban

Szerződések

- A *szolgáltatás orientált architektúrá*ban egy, vagy több komponens (*szolgáltatók*) biztosítja a funkcióknak olyan felületét, amelyet a további komponensek (*fogyasztók*) elérhetnek
  - a szolgáltatás biztosítja interfészének teljes leírását (*szerződés*), így a fogyasztók mindig tisztában lehetnek az elérhető funkciókkal
- A WCF szolgáltatások alapvető szerződése a *szolgáltatásszerződés* (*Service Contract*), amely a biztosított funkciókat ismerteti
  - amennyiben egyedi tartalmat (üzenetek, adattípusok) szeretnénk közvetíteni, további szerződések szükségesek

Kommunikációs megoldások szolgáltatásokban

Szerződések

- *Adatszerződés* (*Data Contract*): a kliens oldalra megosztott összetett adattípus publikus tulajdonságaival (metódusok és mezők nem oszthatóak meg)
  - a szolgáltatott típust a **`DataContract`**, míg a tulajdonságokat a **`DataMember`** attribútummal kell megjelölnünk (felsorolási típus esetén **`EnumMember`**)
  - a primitív típusok előre definiáltak, ezek az úgynevezett implicit adatszerződések
  - az Entity Framework adatmodellek alapértelmezetten adatszerződések is
  - beágyazott, előre nem ismert leszármazott típusok esetén használni kell a **`KnownType`** attribútumot

Kommunikációs megoldások szolgáltatásokban

Szerződések

- pl.:

```
[DataContract] // adatszerződés
class Employee {
    [DataMember] // szolgáltatott tulajdonság
    String Name { get; set;}
    [DataMember]
    Int32 Id { get; set;}
}

[ServiceContract]
public interface IEmployeeService {
    [OperationContract]
    Employee GetEmployeeDetails(Int32 id);
} // a típus felhasználható a metódusban
```

Kommunikációs megoldások szolgáltatásokban

Szerződések

- *Üzenetszerződés*t (*Message Contract*): amennyiben a standard SOAP kommunikáció nem megfelelő az alkalmazás számára, az üzenetszerződés biztosít lehetőséget egyedi üzenetforma megadására
  - az egyéni üzenetszintaxist a **`MessageContract`** attribútummal megjelölt osztállyal adhatjuk meg
  - az üzenet fejlécét a **`MessageHeader`**, törzsét a **`MessageBodyMember`** attribútumokkal megjelölt tulajdonságok alkotják
  - üzenetszerződések csak egy paraméteres műveletben alkalmazhatóak, és a visszatérési érték csak üzenetszerződés lehet, vagy semmi

Kommunikációs megoldások szolgáltatásokban

Szerződések

- pl.:

```
[MessageContract] // üzenetszerződés
class Employee {
    [MessageBodyMember] // törzs
    String Name { get; set;}
    [MessageHeader] // fejléc
    Int32 Id { get; set;}
}

[ServiceContract]
public interface IEmployeeService {
    [OperationContract]
    Employee GetEmployeeDetails(Employee id);
} // csak ezt a típust használhatjuk
```

Kommunikációs megoldások szolgáltatásokban

Szerződések

- *Hibaszerződés* (*Fault Contract*): a szerver oldalon keletkezett hibák pontos feltérképezését tehetjük lehetővé általa
  - alapértelmezés szerint ugyanis a szerveren keletkezett kivételek nem jutnak el a klienshez
  - amennyiben a kivétel üzenetét akarjuk továbbítani a klienshez, használhatjuk a beépített **`FaultException`** típust
  - amennyiben egyedi kivételt definiálunk, azt adatszerződésként tehetjük meg, majd megadhatjuk, melyik műveletek váltják ki az egyedi kivételt a **`FaultContract`** attribútummal

Kommunikációs megoldások szolgáltatásokban

Szerződések

- pl.:

```
[DataContract] // adatszerződés a hibához
class EmployeeException {
    [DataMember]
    Employee Emp { get; set;}
}

[ServiceContract]
public interface IEmployeeService {
    [OperationContract]
    [FaultContract(typeof(EmployeeException))]
    Employee GetEmployeeDetails(Employee id);
} // csak ezt a típust használhatjuk
```

Kommunikációs megoldások szolgáltatásokban

Szolgáltatások példányosítása

- A szolgáltatások példányosítási módját az `InstanceContextMode` tulajdonság szabályozza, lehet:
  - egy példány az összes fogyasztóra (`Single`), azaz minden kliens ugyanaz az objektum szolgál ki, amely meghatározott ideig a memóriában marad
  - egy példány minden fogyasztóra (`PerSession`), azaz minden klienst más objektum szolgál ki, amely meghatározott ideig a memóriában marad
  - egy példány egy hívásra (`PerCall`), azaz minden hívásra új objektum példányosul, majd törlődik a tevékenység végeztével

Kommunikációs megoldások szolgáltatásokban

Párhuzamosság kezelése

- A szolgáltatások és műveleteik párhuzamosan is hívhatóak, és emiatt ügyelnünk kell a kölcsönös kizárásra
  - elsősorban abban az esetben, amikor minden kliens ugyanaz az objektum szolgál ki (`InstanceContextMode.Single`), de statikus mezők, megosztott erőforrások esetén is fontos
  - a kritikus szakaszok mindig kezeljük `lock` utasítással (vagy más módon)
- A probléma megelőzhető, ha *sorosítjuk* a kliensek tevékenységeit, azaz csak egymás után hajthatják végre a tevékenységeket, ehhez a `ConcurrencyMode.Single` beállítást kell végrehajtanunk

Kommunikációs megoldások szolgáltatásokban

Párhuzamosság kezelése

- pl.:

```
[ServiceBehavior(
    InstanceContextMode =
        InstanceContextMode.Single,
    // egy objektum szolgálja ki a klienseket
    ConcurrencyMode = ConcurrencyMode.Single)]
    // de a párhuzamos kéréseket sorosítja
public class EmployeeService : IEmployeeService
{
    // nincs szükségünk kölcsönös kizárásra
    …
}
```

Kommunikációs megoldások szolgáltatásokban

Példa

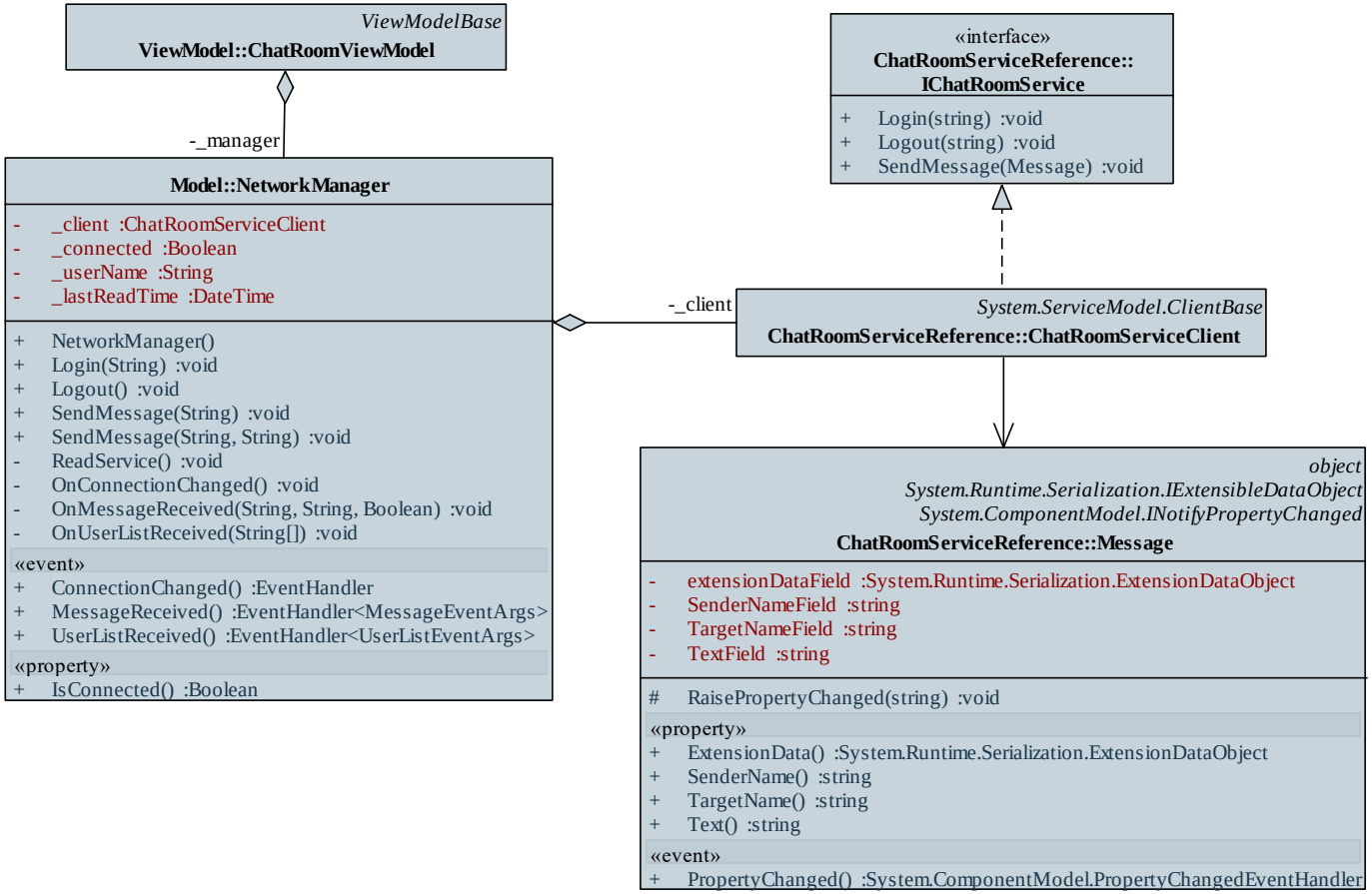
Feladat: Készítsük el a csevegőszobát WCF szolgáltatás segítségével.

- egy szerver (`ChatRoom.ServerService`) és egy kliens (`ChatRoom.Client`) komponensre lesz szükségünk, a szerver egy WCF alkalmazás
- az üzenetet (`Message`) adatszerződésként adjuk meg, a típusára már nem lesz szükség, ezt a megfelelő metódushívások helyettesítik
- mivel a szerver nem kezdeményez üzenetküldést, a kliensnek kell lekérdeznie adott időközönként a felhasználói listát, illetve az üzeneteket (persze elég csak a legutóbbi lekérdezés óta kapottakat), ezt egy külön szálban végezzük

Kommunikációs megoldások szolgáltatásokban

Példa

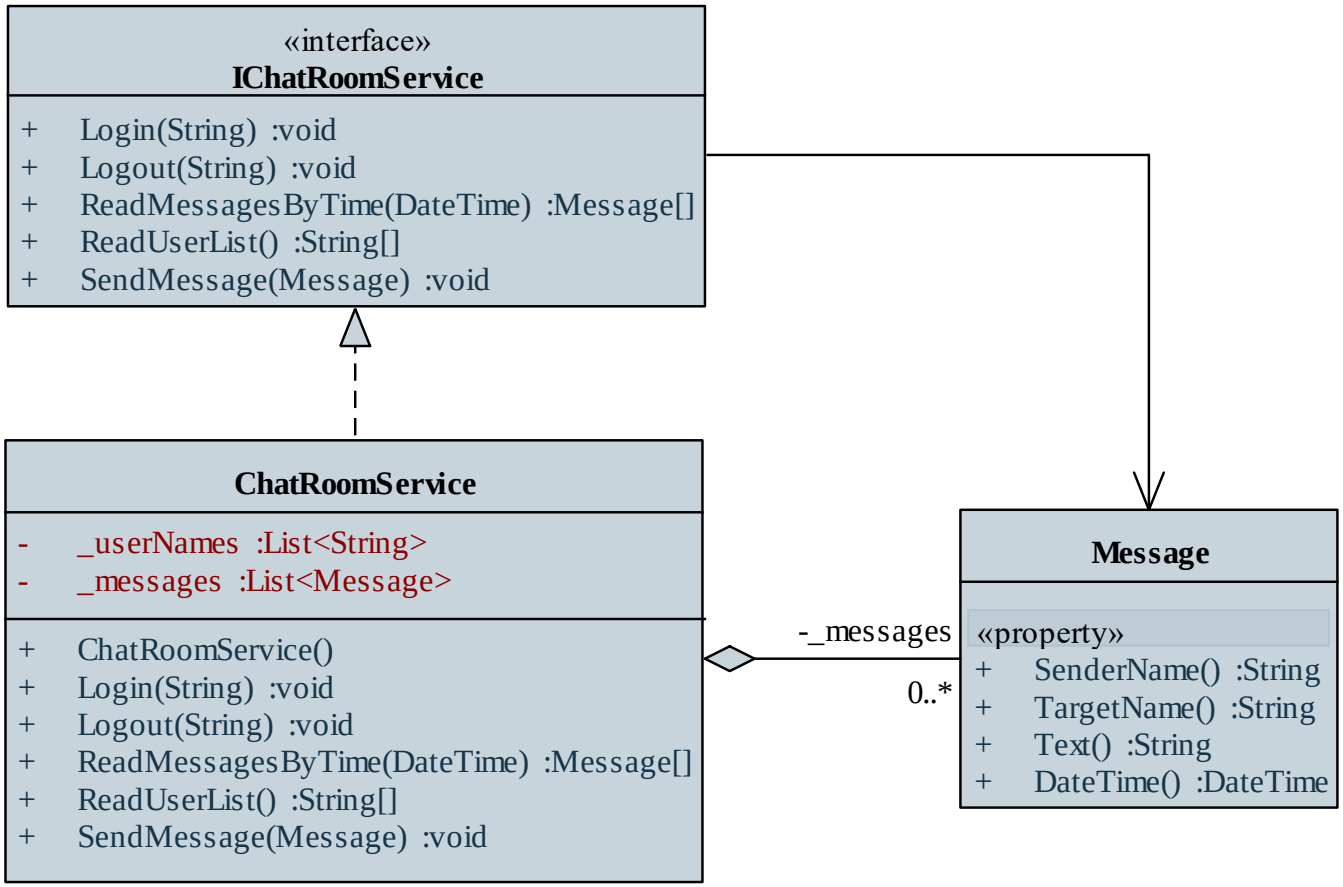
Tervezés (kliens):



Kommunikációs megoldások szolgáltatásokban

Példa

Tervezés (szolgáltatás):



Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`ChatRoomService.svc.cs`*):*

```
…
[ServiceBehavior(
 InstanceContextMode = InstanceContextMode.Single,
 ConcurrencyMode = ConcurrencyMode.Single)]
   // a szolgáltatás egy példányban, soros
   // kiszolgálással fut az összes kliensre
public class ChatRoomService : IChatRoomService{
   …
   public Message[] ReadMessagesByTime(
                                    DateTime time){
      return _messages.Where(message =>
           message.DateTime >= time).ToArray();
   …
```

Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`NetworkManager.cs`*):*

```
…
public void Login(String userName){
     _lastReadTime = DateTime.Now;
     _userName = userName;
     try {
        _client.Login(userName); _connected = true;
        ReadService();
           // ha sikerült a csatlakozás, akkor
           // kezdődhet a kommunikáció
    } catch { _connected = false; }
    OnConnectionChanged();
}
…
```

Kommunikációs megoldások szolgáltatásokban

Kommunikációs protokollok

- Az adatkötéshez a WCF számos kommunikációs protokollt támogat, pl.:
  - *BasicHttpBinding*: webszolgáltatásnak megfelelő kommunikáció alapszolgáltatásokkal
  - *WSHttpBinding*: tranzakciókat és biztonságot támogató webszolgáltatás
  - *WSDualHttpBinding*: kétirányú kommunikációt biztosító webszolgáltatás
  - *NetPeerTcpBinding*: peer-to-peer alapú kommunikáció
  - *NetTcpBinding*: WCF alkalmazások közötti kommunikáció tetszőleges gépek között

Kommunikációs megoldások szolgáltatásokban

Kommunikációs protokollok

- A kommunikációs protokollt a konfigurációban állíthatjuk be (kliens és szerver oldalon), pl.:

```
<system.serviceModel>
    <!-- definiáljuk a szolgáltatást -->
    <services>
      <service name="EmployeeService">
        <endpoint binding="wsDualHttpBinding"
                  contract="IEmployeeService">
        </endpoint> <!-- speciális kötés -->
      </service>
    </services>
    <behaviors>
    …
</system.serviceModel>
```

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- A WCF lehetőséget biztosít, hogy a szolgáltatót aktívvá tegyünk, ekkor ő is kezdeményezhet kommunikációt *visszahívás* (*callback*) formájában
  - a visszahívás csak speciális kötésekkel működik (**`wsDualHttpBinding`** és **`netTcpBinding`**)
  - a visszahívás lényegében egy reakció a kliens kérésére, amely a válasz után bármikor végrehajtható

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- A visszahíváshoz is tartozik egy szerződés, interfészként megadva
  - pl.:

```
public interface IEmpCallback {
    [OperationContract] // visszahívó művelet
    void RaisePayment(Int32 amount);
}
```

  - a visszahívást csak bizonyos időtartamig lehet kezdeményezni a hívást követően, ez kiküszöbölhető az egyirányúsírás (`IsOneWay`) beállításával, pl.:

```
[OperationContract(IsOneWay = true)]
    // egyirányú visszahívó művelet
void RaisePayment(Int32 amount);
```

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- Az interfészt meg kell adnunk visszahívási attribútumként a szolgáltatás szerződésben a **`CallbackContract`** tulajdonsággal
  - pl.:

```
[ServiceContract(CallbackContract =
                    typeof(IEmpCallback))]
   // szolgáltatás szerződés visszahívással
public interface IEmployeeService { … }
   // magát a szerződés tartalmát nem kell
   // módosítani
```

  - a műveletek eléréséhez a szolgáltatásban megvalósítjuk az interfészt a **`GetCallbackChannel<…>()`** művelethívással, ahol a sablonban adjuk át a szerződést

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- pl.:

```
public class EmployeeService : IEmployeeService
{
    …
    // példányosítjuk a visszahívó csatornát
    IEmpCallback cb = OperationContext.Current
                .GetCallbackChannel<IEmpCallback>();

    // majd ezen keresztül hívhatjuk a kliens
    // metódusait
    cb.RaisePayment(…);
       // a művelet a kliensen fut le
    …
}
```

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- A kliensnek is támogatnia kell a visszahívást, és meg kell valósítania a visszahívás szerződését
  - a kliens osztályt a **`DuplexClientBase`** típusból kell származtatni (a visszahívás támogatásához)
  - példányosítani kell egy fogadó objektumot, amely megvalósítja a visszahívás interfészét, amelyet tovább kell adni a kliensnek az **`InstanceContext`** típusba csomagolva
  - pl.:

```
class EmpCallback: IEmpCallback { … }
   // kliens oldalon létrehozott típus, amely
   // megvalósítja a visszahívás szerződését
```

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- pl.:

```
[ServiceBehavior(…)] // szolgáltatás kliens
class EmployeeServiceClient
    : DuplexClientBase<IEmployeeService>,
      IEmployeeService {
      // támogatja a visszahívást

    public EmployeeServiceClient(InstanceContext
        context) : base(context) { … }
        // a konstruktor egy InstanceContext
        // példányt fogad
    …
};
```

Kommunikációs megoldások szolgáltatásokban

Visszahívásos szolgáltatások

- pl.:

```
EmpCallback callback = new EmpCallback();
    // fogadó objektum, amely megvalósítja az
    // interfészt
InstanceContext context =
    new Instancecontext(callback);
    // átadjuk a fogadó objektumot

EmployeeServiceClient client =
    new EmployeeServiceClient(context);
    // a szolgáltatás így megkapja a fogadó
    // objektumot, innentől a visszahívás a
    // callback.RaisePayment(…) művelethívást
    // fogja futtatni
```

Kommunikációs megoldások szolgáltatásokban

Példa

Feladat: Módosítsuk a csevegőprogramot úgy, hogy visszahívások segítségével történjen a kommunikáció.

- csökkenti a terhelést, mivel nem kell adott időközönként lekérdezni a szervert, hanem minden változás esetén a szerver elküldi a módosításokat (üzenet esetén elég az új üzenetet továbbítani)
- a megvalósításhoz **`WSHttpBinding`** kötést használunk
- megvalósítjuk a visszahívás szerződését (**`IChatRoomCallbackService`**), amely lehetőséget ad üzenetek és felhasználói lista küldésére
- a szerver szolgáltatásból így kikerülnek metódusok, amelyek a kliensnél jelennek meg

Kommunikációs megoldások szolgáltatásokban

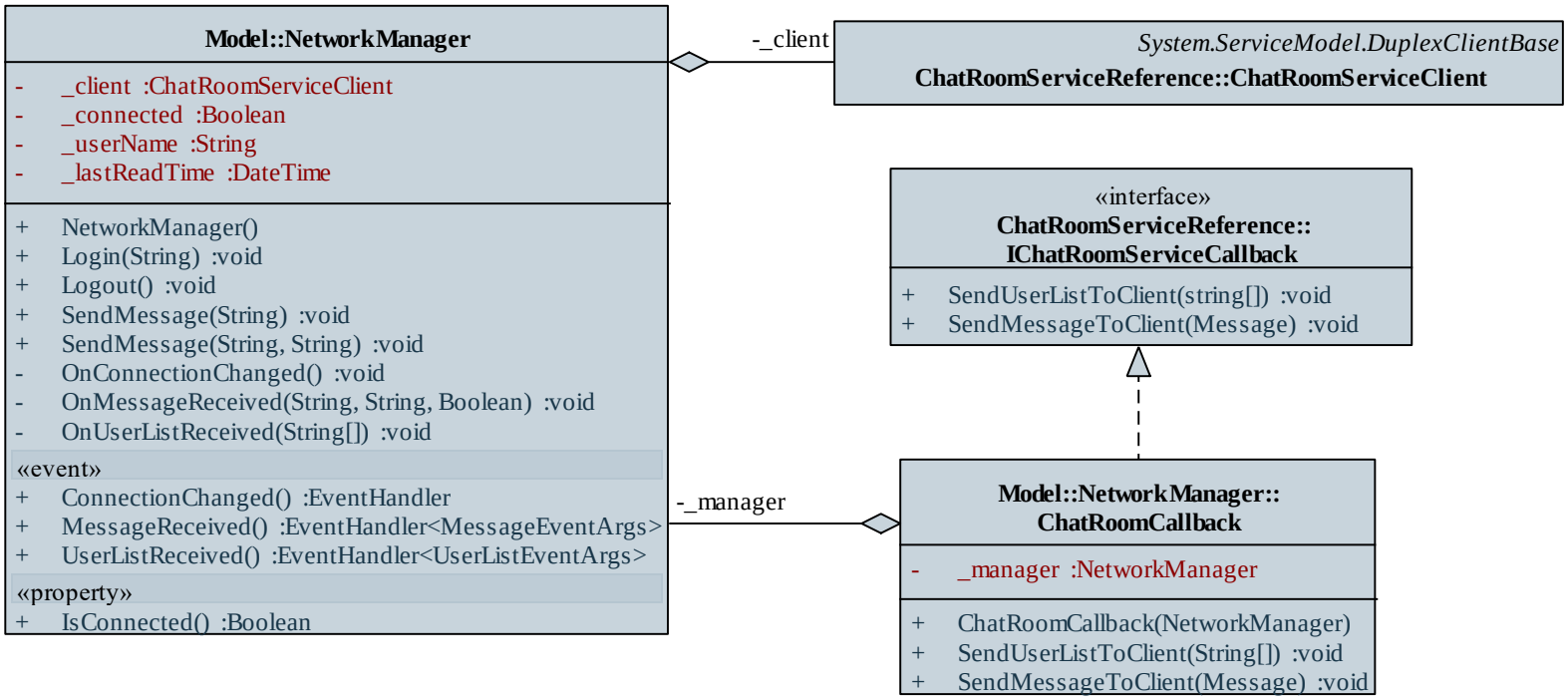
Példa

- a szerveren kiküszöböljük a párhuzamosítást sorosítással (`ConcurrencyMode.Single`)
- a szerveren a visszahívási csatornákat egy listában (`_userCallBackServices`) tároljuk, így utólag elérhetjük az összes kliens visszahívóját, és küldhetünk üzenetet
- a kliensnek sem kell párhuzamos szálban tevékenykednie (és a szinkronizációval sem kell törődnie), a visszahívásra kell csupán reagálni egy megfelelő kezelő objektumban (`ChatRoomCallback`), amelyet beágyazott típusként valósítunk meg a `NetworkManager`-en belül, így könnyen elérheti annak privát műveleteit

Kommunikációs megoldások szolgáltatásokban

Példa

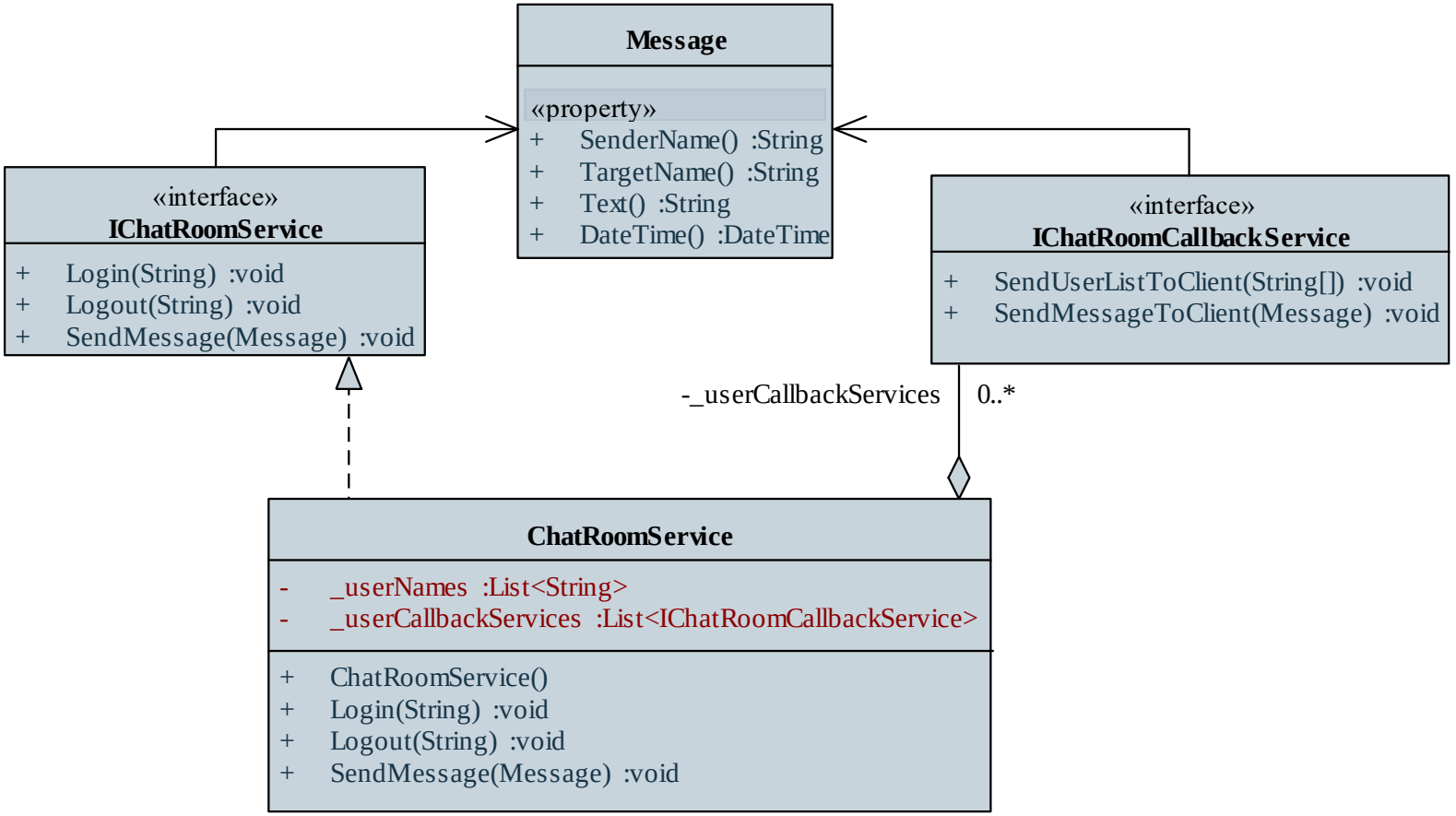
Tervezés (kliens):



Kommunikációs megoldások szolgáltatásokban

Példa

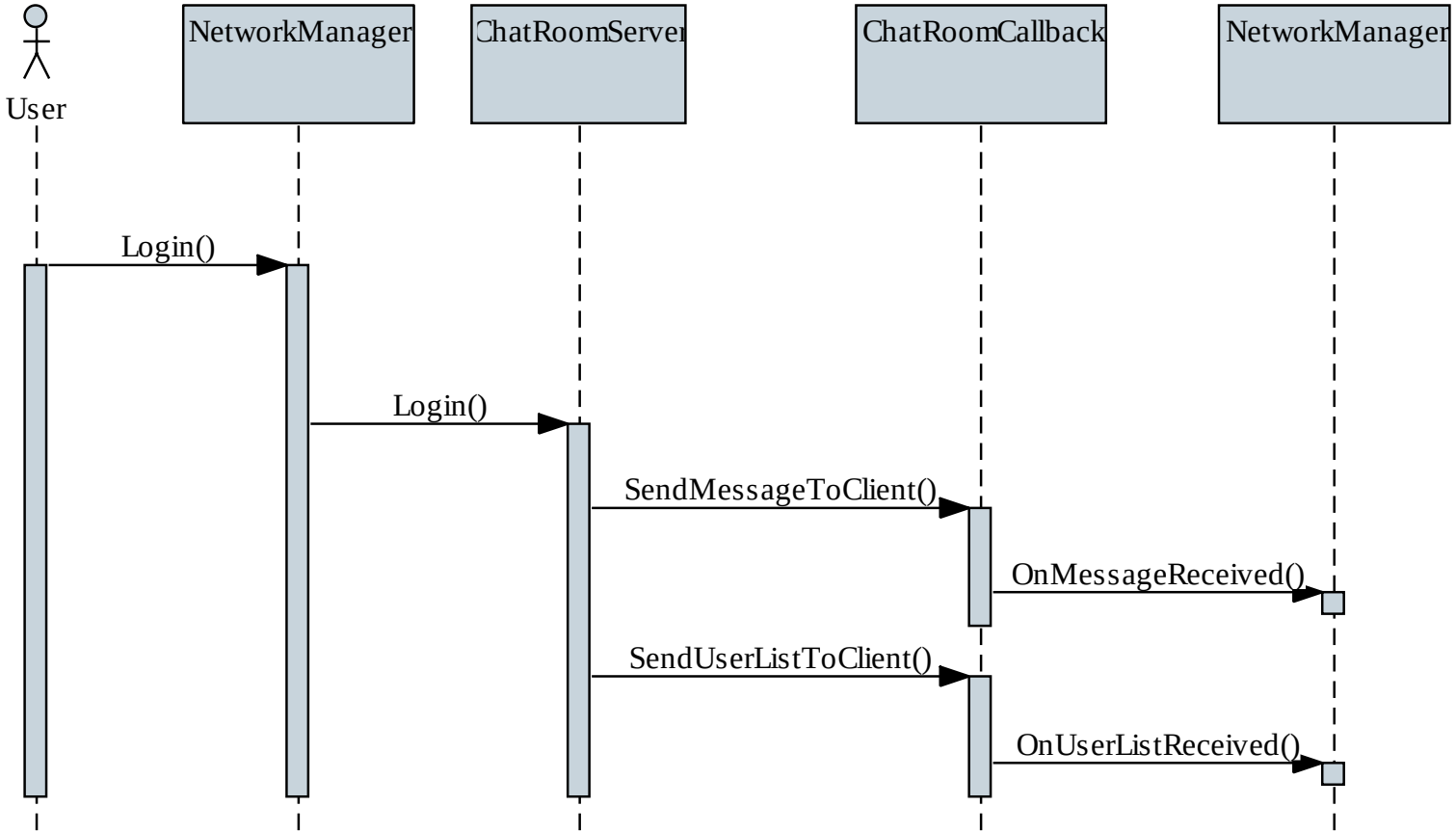
Tervezés (szolgáltatás):



Kommunikációs megoldások szolgáltatásokban

Példa

Tervezés:



Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (**web.config**):

```
…
<service behaviorConfiguration="ServiceBehavior"
         name="ELTE.Net.ChatRoom.
               ServerService.ChatRoomService">
   <endpoint binding="wsDualHttpBinding"
             contract="ELTE.Net.ChatRoom.
                ServerService.IChatRoomService">
      <!-- kétirányú webszolgáltatás alapú
           kommunikációt használunk -->
          …
   </endpoint> …
 </service>
…
```

Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`NetworkManager.cs`*):*

```
public class NetworkManager {
    private class ChatRoomCallback :
        IChatRoomServiceCallback {

        private NetworkManager _Manager;
        …
        public void SendUserListToClient(String[]
            userNames){
            _Manager.OnUserListReceived(userNames);
                // a menedzseren keresztül meghívja a
                // megfelelő eseménykiváltást
        }
        …
```

Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`ChatRoomService.svc.cs`*):*

```
…
_UserCallbackServices.Add(
   OperationContext.Current.GetCallbackChannel
                        <IChatRoomCallbackService>());
   // létrehozzuk a neki megfelelő visszahívó
   // csatornát és felvesszük a többiek közé

//  frissítjük a felhasználói listát
foreach (IChatRoomCallbackService
        callbackService in _UserCallbackServices){
   callbackService.SendUserListToClient(
                                _UserNames.ToArray());
} …
```

Kommunikációs megoldások szolgáltatásokban

Fogyasztók kezelése

- A szolgáltatásokat általában adott periódusra veszik igénybe a fogyasztók, a periódus kezdetekor a fogyasztó *feliratkozik* (*subscribe*) a szolgáltatásra, a végén *leiratkozik* (*unsubscribe*)
  - a leirakoztatás lehet automatikus (pl. inaktivitás miatt)
  - WCF-ben ez egyszerű metódusokkal is megvalósítható, illetve kliensenként (`PerSession`) történő kezeléssel

Kommunikációs megoldások szolgáltatásokban

Munkafolyamatok megvalósítása

- A *munkafolyamat* (*session*) alapú kezelés így lehetővé teszi egy kliens folyamatos követését, állapotának felügyeletét
  - a szerződésnél megadható, hogy csak munkafolyamaton keresztül használhatóak a műveletek (**`SessionMode.Required`**)
  - minden műveletre megadható, hogy inicializálja a munkafolyamatot (**`IsInitiating`**), terminálja (**`IsTerminating`**), vagy egyik sem
  - az **`InstanceContextMode.PerSession`** beállítás munkafolyamatonként példányosít egy klienst
  - csak bizonyos kötésekkel vehető igénybe (pl. **`WSHttpBinding`**, **`NetTCPBinding`**)

Kommunikációs megoldások szolgáltatásokban

Munkafolyamatok megvalósítása

- Pl.:

```
[ServiceContract(SessionMode =
     SessionMode.Required)]
   // elvárjuk a munkafolyamat alapú kezelést
public interface IEmployeeService {
   [OperationContract(IsInitializing = true)]
   void Login(String name);
   // munkafolyamat nyitó művelet

   [OperationContract(IsInitializing = false,
                      IsTerminating = true)]
   void Logout();
   // munkafolyamat záró művelet
   …
```

Kommunikációs megoldások szolgáltatásokban

Példa

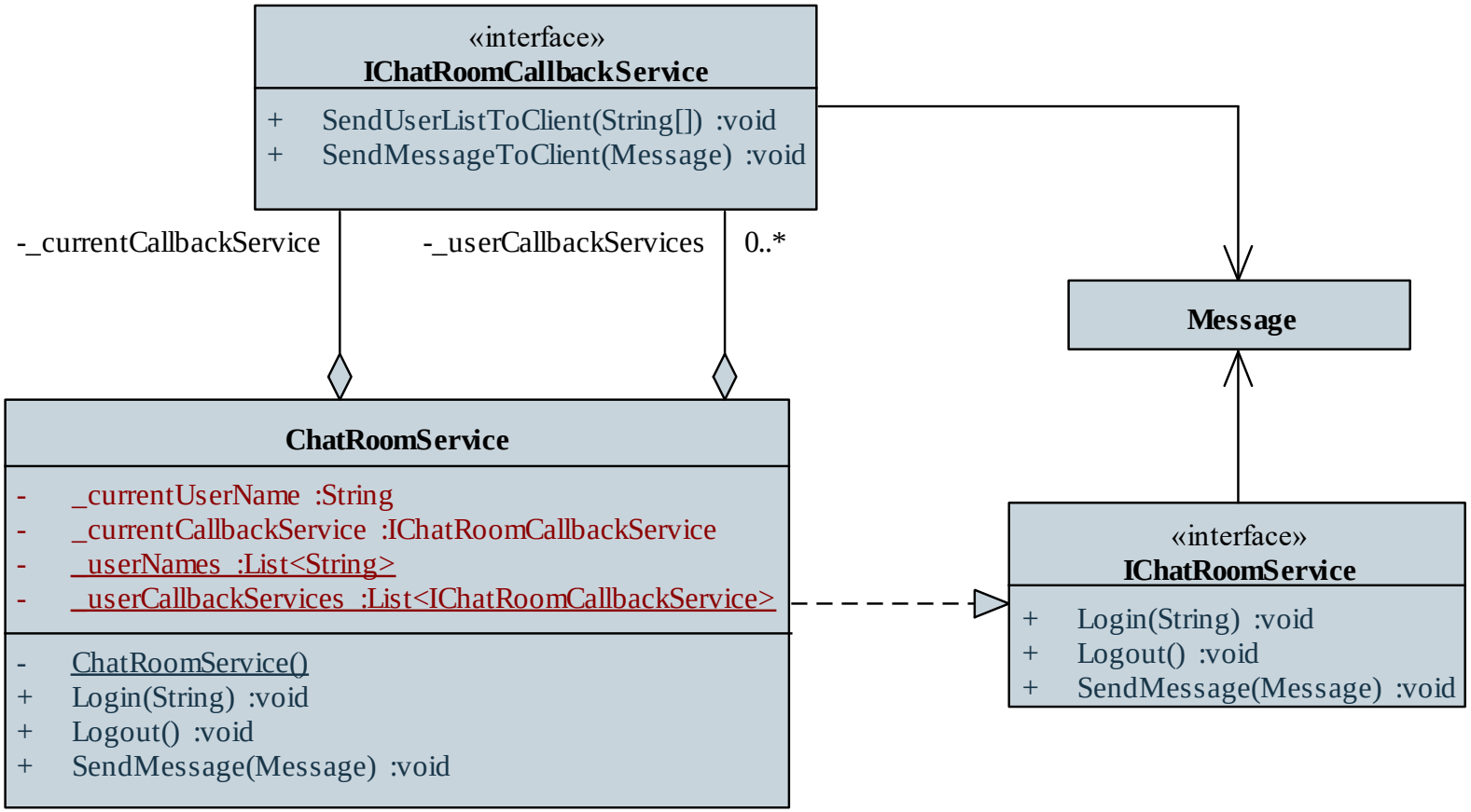
Feladat: Módosítsuk a csevegőprogramot úgy, hogy munkafolyamat alapú legyen a kommunikáció.

- a kliens a `Login(…)` művelettel indítja a munkafolyamatot és a `Logout()` művelettel zárja (így már nem kell kijelentkezéskor megadni a felhasználónevet)
- a szolgáltatás munkafolyamat szinten eltárolja a felhasználónevet és a visszahívó csatornát a későbbi azonosításhoz, a felhasználók, illetve csatornák listája osztályszintű lesz, így minden kliens hozzáférhet
- az osztályszintű mezőkre biztosítani kell a kölcsönös kizárást

Kommunikációs megoldások szolgáltatásokban

Példa

Tervezés:



Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`IChatRoomService.cs`*):*

```
[ServiceContract(CallbackContract = …,
    SessionMode = SessionMode.Required)]
    // elvárjuk a munkafolyamat alapú kezelést
public interface IChatRoomService {
    [OperationContract(IsOneWay = true,
     IsInitiating = true, IsTerminating = false)]
    void Login(String userName);
       // indítja a munkafolyamatot

    [OperationContract(IsOneWay = true,
     IsInitiating = false, IsTerminating = true)]
    void Logout(); // lezárja a munkafolyamatot
…
```

Kommunikációs megoldások szolgáltatásokban

Példa

Megvalósítás (`ChatRoomService.svc.cs`*):*

```
[ServiceBehavior(InstanceContextMode =
    InstanceContextMode.PerSession,
    ConcurrencyMode = ConcurrencyMode.Single)]
// a szolgáltatás külön példányban fut az összes
// kliensre, sorosítva (egy kliens nem végezhet
//  párhuzamos tevékenységet)
public class ChatRoomService : IChatRoomService{
    private String _CurrentUserName;
       // eltároljuk a felhasználónevet
    private IChatRoomCallbackService
       _CurrentCallbackService; // és a visszahívó
       // csatornát az adott kliensre
…
```