Budapesti Műszaki és Gazdaságtudományi Egyetem
Intelligens rendszerfelügyelet (BME VIMIA370)

Házi feladat 3-D

Felügyeletre tervezés

Kovács Balázs (HZFWTN)
2011. április 26.

1 Környezet

A kiíráshoz mellékeljük egy egyszerű, C#-ban megvalósított kulcs-érték párokat tároló kiszolgáló és egy hozzá való kliens forráskódját (az alkalmazás 4.0-ás .NET Frameworkhöz készült az ingyenes Microsoft Visual C# 2010 Express Edition eszközzel).

A fejlesztők „elfelejtettek" fejlesztői és üzemeltetői dokumentációt készíteni az alkalmazáshoz. A szervert a létrehozandó adatbázis nevével kell elindítani. A szerver az inicializálás után a kiírt URL-en fogadja a kliensek kéréseit. Ezen felül a szerver felületén tudunk kulcs-érték párokat hozzáadni az „adatbázishoz". A kliensnek át kell adni a szerver URL-jét paraméterként. Ezek után lehet kulcsszavakat kérdezni a szervertől. Ha az szerepel az adatbázisában, akkor a hozzá tartozó értéket visszaadja, egyébként hibaüzenettel tér vissza.

2 Modellezés

Hogy jobban megértsük az alkalmazás működését, a forráskód tanulmányozásával készítsen egy UML modellt (tipikusan komponens vagy osztálydiagramokkal), ami ábrázolja a rendszer főbb részeit és azok kapcsolatait. A modell tartalmazza, hogy melyik komponens milyen adatot tárol, hogyan kommunikálnak egymással a komponenseink, tipikus esetben melyikből hány darab fut, stb.

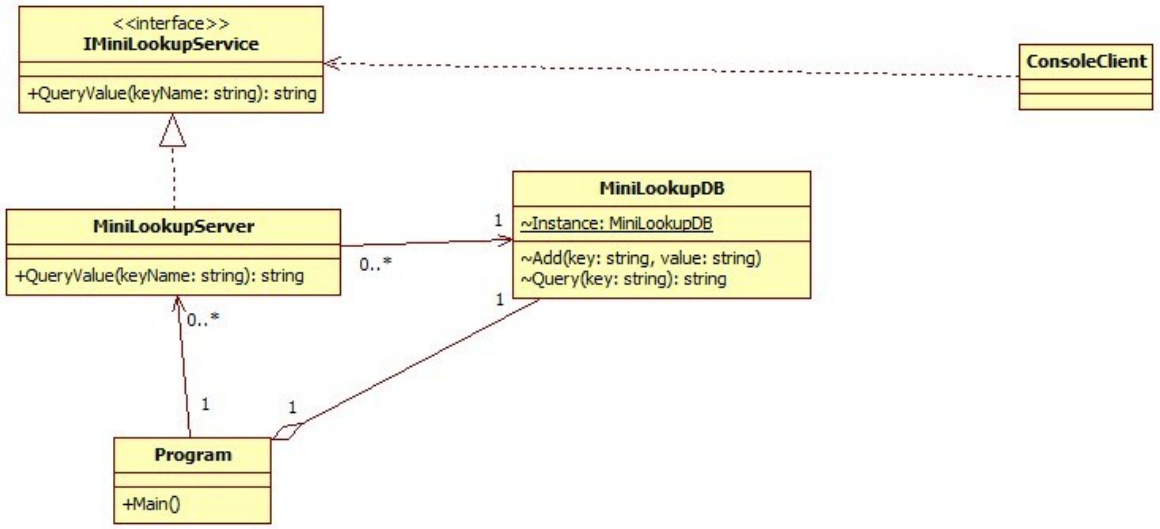
Készítse el ezen kívül a kiszolgáló komponens viselkedését leíró dinamikus modellt is UML állapotgépek formában.

2.1 Használt UML szerkesztőprogram

A modellezési feladatokat a StarUML segítségével készítettem el.

2.2 Statikus modell

2.2.1 Osztálydiagram

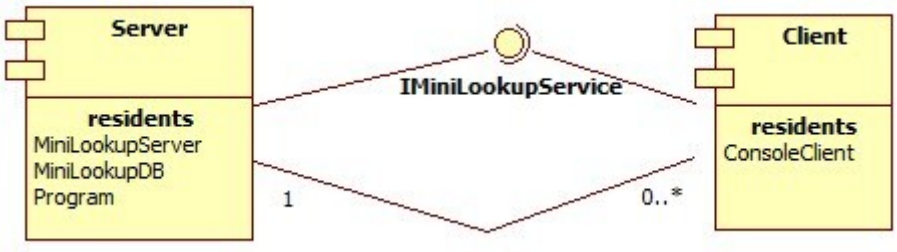


Magyarázat:

A programkód tanulmányozásával felvettem a definiált osztályokat és interfészeket. A szerveroldal a baloldalon látható, a kliens pedig a jobboldalt. A szerver futása során létrehoz egy példányt a MiniLookupDB-ből (ez Singleton mintát valósít meg, vagyis csak egy példány lehet

belőle), majd várja a kliens kéréseit. A MiniLookupServer osztály megvalósítja az IMiniLookupServer interfészt, amit WCF segítségével kiajánlunk. Erre az interfészre "csatlakozik rá" a kliens, vagyis ha egy kérés érkezik a klienstől a WCF rendszer legenerál egy új MiniLookupServer objektumot, ami kiszolgálja a kérést. Természetesen ez alatt a szerveroldalon végig lehet használni az adatbázist, fel lehet tölteni új értékekkel.

2.2.2 Komponens diagram

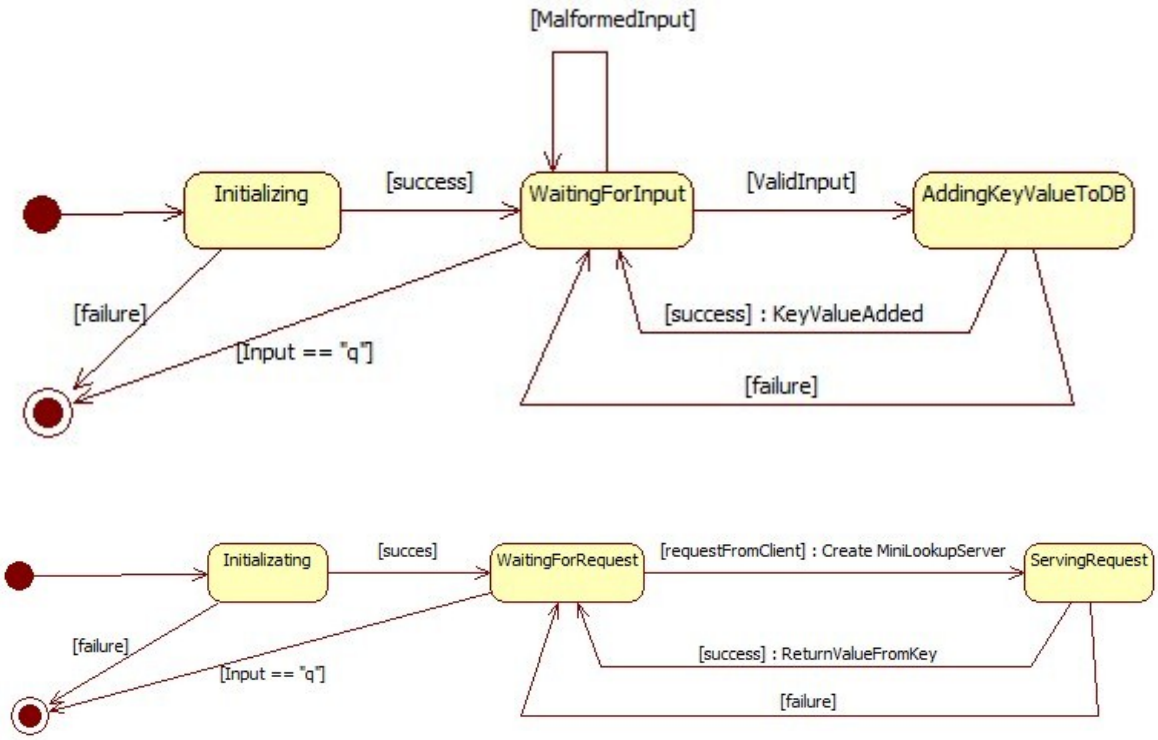


Magyarázat:

Ez a diagram jobban leírja a kliens-szerver kapcsolatot. Megjelenik mindegyik komponensben, hogy melyik osztályok alkotják (residents). Az asszociációt azért tettem be, hogy látszódjon tipikusan hány darab fut az egyes komponensekből.

2.3 Állapot modell

A szerverhez kétféle állapotmodell tartozik. Az első az, amikor a szerverkarbantartó feltölti az adatbázist elemekkel, a második pedig az, amikor a szerver kiszolgálja a kliens kérését. A naplózási hívásokat természetesen nem raktam bele a modellbe, hiszen azokat csak az ez utáni feladatokban dolgoztam ki.



3 Felügyeleti modell elkészítése

Megfelelő érveléssel alátámasztva sorolja fel, hogy a kiszolgáló komponensnek milyen, a működésével kapcsolatos eseményeket kellene rendszerfelügyeleti célokra elérhetővé tennie! Az eseményekhez legalább a következő adatokat adja meg:

- esemény azonosítója (integer),
- esemény súlyossága (a .NET System.Diagnostics.TraceEventType felsorolás súlyossági szintjei szerint),
- esemény szövege,
- az esemény definíciója (annyira formálisan és részletesen, hogy az további pontosítás nélkül implementálható legyen).

Legalább **nyolc** esemény definiálását várjuk el, ezek között kell lennie Critical, Error vagy Warning, Information és Verbose súlyosságú eseménynek is.

3.1 Események összegyűjtése

ID	Súlyosság	Szöveg/Definíció
101	Information	"The MiniLookupServer service is ready at {0}" Az alkalmazás sikeresen befejezte a bemeneti paraméterek feldolgozását, létrehozta a szükséges objektumokat és a szerver szolgáltalás sikeresen elindult a {0} paraméterben írt címen.
102	Information	"MiniLookupDB created for database {0}" Az adatbázist sikeresen létrehoztuk a {0} paraméterben írt néven.
103	Information	"MiniLookupServer constructor called" Egy új szerverobjektum példány jött létre, meghívták az osztály konstruktorát.
201	Critical	"Lookup database name not supplied." A szerveralkalmazás indításánál nem adtuk meg az adatbázis nevét.

202	Critical	"Lookup database name not valid." A szerveralkalmazás indításánál nem adtuk meg érvényes adatbázisnevet.
203	Critical	"Could not initialize MiniLookupServer service. Reason: {0}" Valamilyen okból nem lehet inicializálni a szolgáltatást, ezt az okot a {0} paraméterben láthatjuk.
301	Error	"Malformed input." Nem a kért formátumnak megfelelő módon adtuk meg a bevinni kívánt kulcs-érték párt.
901	Verbose	"New query received from client for key {0}" Új lekérdezés érkezett a {0} paraméterben írt nevű kulcshoz tartozó értékre.
902	Verbose	"Added value '{0}' for key '{1}'" Az adatbázisunkhoz új kulcs-érték párt adtunk. A kulcs az {1} paraméterben, az érték pedig a {0} paraméterben olvasható.
903	Verbose	"MiniLookupServer service is closed." A szolgáltatást lezártuk. (Ez a program futása végén van)

3.1.1 Magyarázat

Minden olyan esemény, ami a program futását megszakító hiba, felvettem Critical súlyossággal. Ezeket kiemelkedően fontos naplózni. Ez után olyan hibákat kerestem, amit nem szakítja meg a program futását, ezek kerültek az Error kategóriába. A hibát nem okozó, de fontos eseményeket Information kategóriába soroltam, a maradék valamilyen hasznos információval rendelkező eseményt pedig a Verbose kategóriába.

Az information típusú események 1xx-es ID-vel rendelkeznek. A critical és warning/error típusú eseményeket szétválasztottam, hiszen a critical típusú események nem javítható hibát jeleznek, így fontos, hogy jól elkülönítsük őket ID szinten is. Az error és warning típusú eseményeket nem különböztetem meg, mert a szolgáltatás futása egyiknél sem áll meg, a hibát valahogy kezelni tudjuk. A verbose típusú üzeneteknek azért adtam 9-el kezdődő ID-t, mert ezeket csak ritkán használjuk, így ezeket egyértelműen el akartam különíteni. Az események szövege sokszor paraméterezett így több információt tudunk átadni velük.

4 Naplózás implementálása

Microsoft Enterprise Library 5.0 – Logging Application Block instrumentáció hozzáadásával és a szükséges egyéb módosításokkal érje el, hogy a kiszolgáló fent definiált eseményei naplózva legyenek! Felhívjuk rá a figyelmét, hogy a kiadott kódban a hibakezelés szándékoltan részleges és elnagyolt, annak rendbetétele a feladat részét képezi.

A feladat megoldásának része az a mérnöki érvelés, hogy hány különböző helyre és milyen logika szerint naplóz az alkalmazás. Használjon legalább **két** különböző típusú Listenert!

4.1 Környezet beállítása

A kért programokat feltelepítettem: FxCop, StyleCop és az Enterprise Library.

4.2 Eseményekhez tartozó naplóüzenetek bevitele erőforrás bejegyzésekbe

ID	Az erőforrás bejegyzés neve
101	ServerServiceReadyLogMessage
102	DatabaseCreatedLogMessage
103	ServerConstructorCalled
201	DatabaseNameNotSuppliedLogMessage
202	DatabaseNameNotValidLogMessage
203	ServerServiceInitFailLogMessage

301	MalformedInputLogMessage
901	NewQueryFromClientLogMessage
902	AddedKeyValueToDatabaseLogMessage
903	ServerServiceIsClosedLogMessage

Az eseményekhez tartozó szövegeket a projekthez tartozó erőforrásokhoz adtam. Ezen kívül létrehoztam egy struktúrát LogEventIds néven, amelyben az eseményekhez tartozó ID-eket tárolom. Ezeknek nevei megegyeznek az erőforrás bejegyzések neveivel a LogMessage vég nélkül.

4.3 Naplózó hívások beépítése a kódba

A programozás könnyítésére és a program átláthatóbbá tételére létrehoztam egy segédfüggvényt, ami ellenőrzi, hogy a naplózás be van-e kapcsolva és megfelelően kitölti a LogWriter Write függvényét.

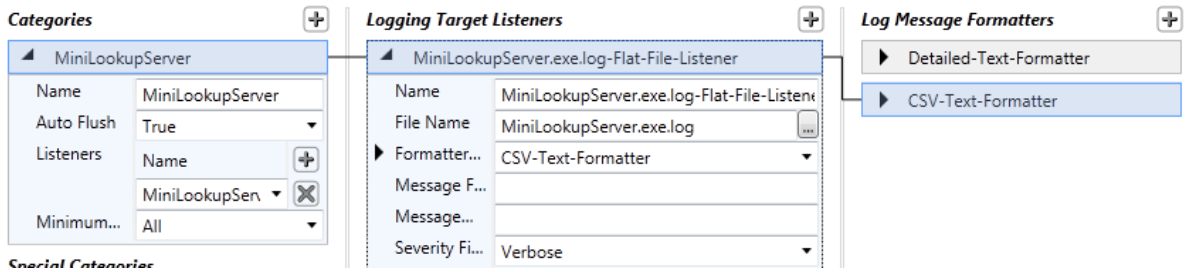
```
internal static void Log(string message, int eventId, TraceEventType severity,
params object[] args)
```

Egy kis trükkel azt is megoldottam, hogy az erőforrásokban tárolt szövegekben paramétereket lehessen elhelyezni, amit ezen Log függvény meghívása során meg lehet adni. Létrehoztam egy LogCategories nevű struktúrát, amiben a kategória neveket tárolom. A fent leírt Log függvény alapértelmezetten a Program kategóriát választja, de van egy másik Log függvény is, ahol beállíthatjuk, hogy melyik kategóriát szeretnénk használni a naplóbejegyzés kiírásához.

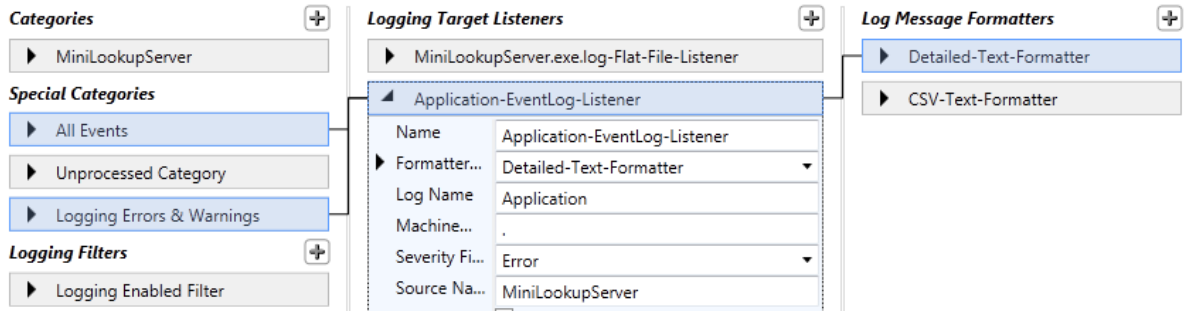
Ezek után már csak a megfelelő helyeken kell meghívni a Log függvényt.

4.4 Naplózás beállítása

Az alkalmazás számára egy Category-t definiáltam MiniLookupServer néven. Ez minden üzenetet átenged. Ennek kimenetét a MiniLookupServer.exe.log-Flat-File-Listener nevű Listener-re kapcsoltam, ami minden eseményt megjelenít, ami legalább Verbose súlyosságú. Ez a Listener a MiniLookupServer.exe.log nevű fájlba naplóz a CSV-Text-Formatter formázó segítségével. Ez a naplófájl inkább az alkalmazásfejlesztőknek, tesztelőknek hasznos. A jobb átláthatóság kedvéért álljon itt egy kép:



Létrehoztam egy másik Listener-t is Application-EventLog-Listener néven, ami a Windows Event Log-ba naplózza az All Events Category és a Logging Errors & Warnings eseményeit. Ez csak az Error és annál nagyobb súlyosságú eseményeket naplózza, legfőképpen a program felhasználói számára, akik kényelmesen megtekinthetik az alkalmazáshoz tartozó naplóbejegyzéseket valamilyen hiba esetén. A naplózáshoz a Detailed-Text-Formatter-t használja, ami az eseményről minden részletet megjelenít.



4.5 Hibakezelés javítása

A program hibakezelése nem volt alapvetően hiányos. Néhány helyen kiegészítettem a hibák felhasználóbarát megfogalmazásával, néhány try-catch blokkal. A legtöbb helyen a sima Exception típusú kivétel elkapása elég, mivel a kivétel üzenet része elég részletesen leírja a hiba okát.

4.6 Tesztelés

Az alkalmazást olyan bemenetekkel kívánom tesztelni, amik teljesen lefedik az implementált naplózási lehetőségeket. A parancssor kimenetét nem másolom ide, hiszen a feladat a naplózás megvalósítása volt, azt akarjuk tesztelni, hogy ez megfelelő-e. Minden teszteset előtt a naplófájl és a Windows Event Log-ot kitöröltem, hogy csak az adott teszteset eseményét lássuk.

A Client programot nem indítjuk el, először csak a Server-t teszteljük.

4.6.1 Teszteset

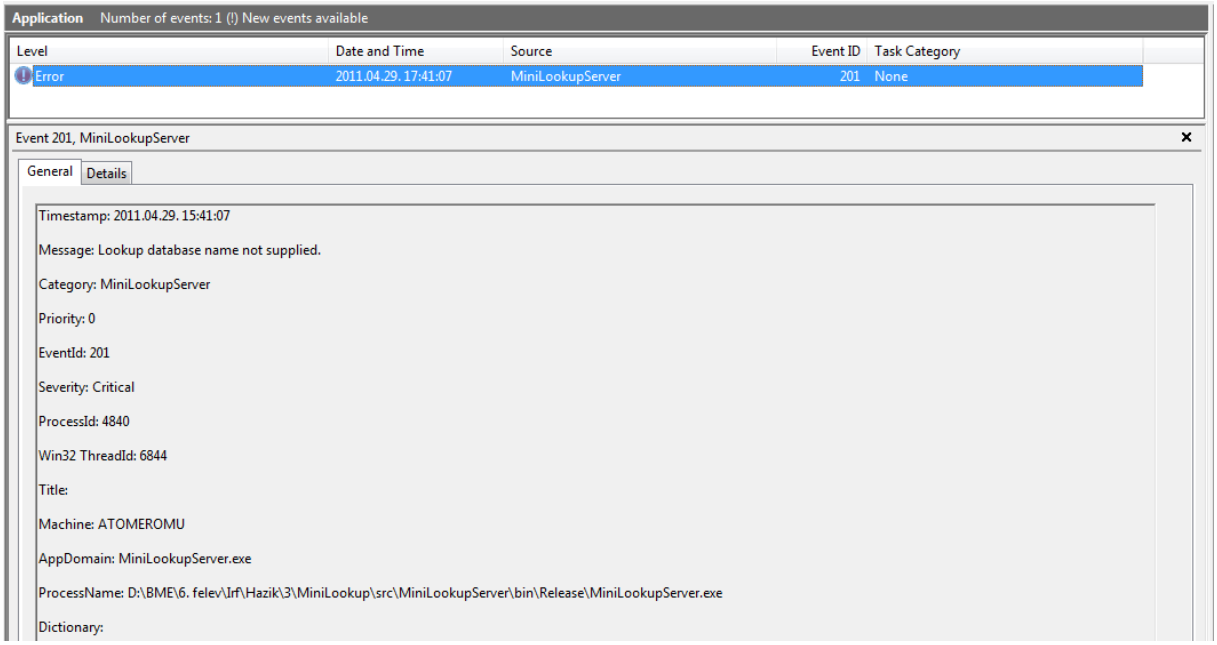
Nem adunk meg adatbázist a szerveralkalmazásnak. Tehát a parancssorban a következő módon hívjuk:

```
MiniLookupServer.exe
```

A naplófájl tartalma:

```
2011-04-29,15:41:07.096,"Lookup database name not
supplied.",MiniLookupServer,201,Critical,4840,6844
```

A Windows Event Log tartalma:



4.6.2 Teszteset

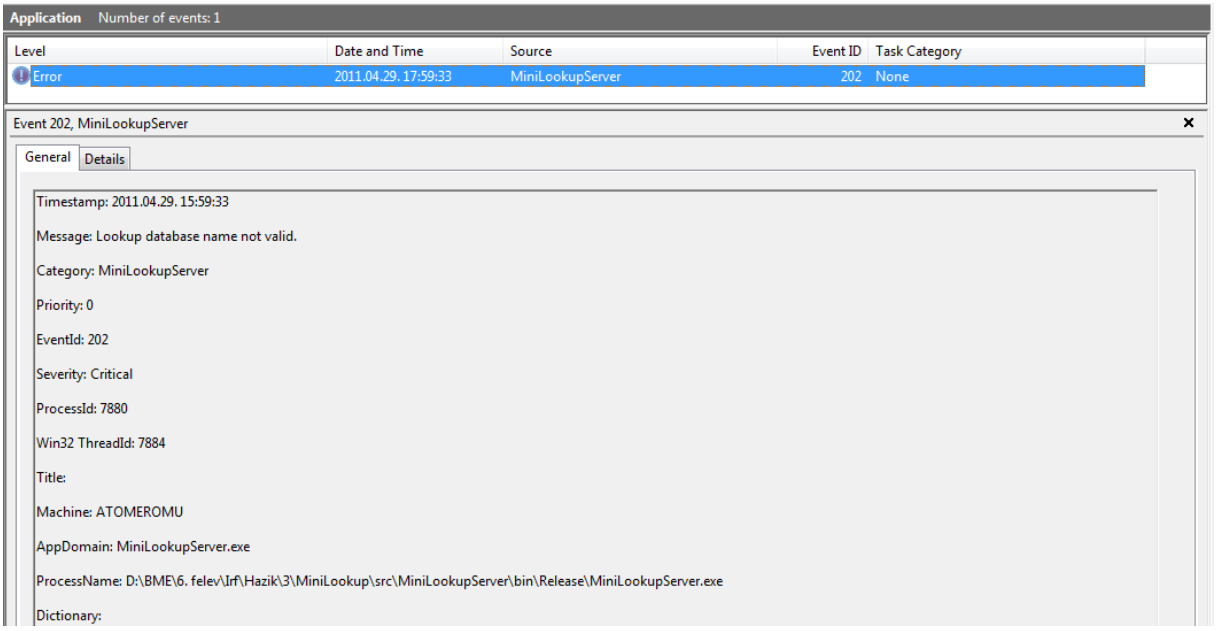
Nem érvényes adatbázist adunk meg az alkalmazásnak. Tehát a parancssorban a következő módon hívjuk:

```
MiniLookupServer.exe 1111
```

A naplófájl tartalma:

```
2011-04-29,15:59:33.671,"Lookup database name not
valid.",MiniLookupServer,202,Critical,7880,7884
```

A Windows Event Log tartalma:



4.6.3 Teszteset

Megfelelő adatbázisnevet adunk meg az alkalmazásunknak, ezentúl mindig ezt az adatbázisinevet fogjuk használni. Tehát a parancssorban a következő módon hívjuk:

```
MiniLookupServer.exe testsrv
```

Ezek után kétszer rossz formátummal adjuk meg a kulcs-érték párokat, majd egyszer jól. Ez után elindítjuk a kliensprogramot és lekérdezünk egy értéket egy kulcs alapján, majd kilépünk a szerveralkalmazásból.

Tehát ezeket írjuk be a parancssorba a szervernél:

```
a
a,
a,b
q
```

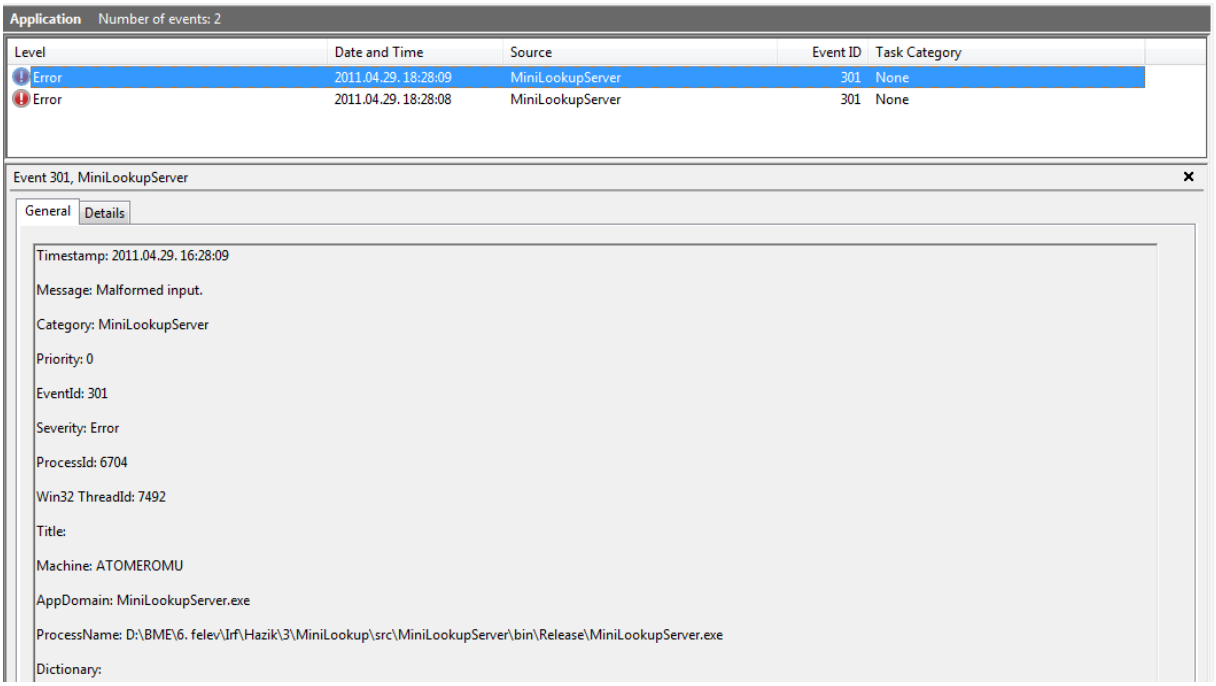
És ezeket a kliensnél:

```
MiniLookupClient.exe http://localhost:8080/testsrv
a
```

A naplófájl tartalma:

```
2011-04-29,16:28:03.911,"MiniLookupDB created for database
testsrv",MiniLookupServer,102,Information,6704,7492
2011-04-29,16:28:03.998,"The MiniLookupServer service is ready at
http://localhost:8080/testsrv",MiniLookupServer,101,Information,6704,7492
2011-04-29,16:28:08.666,"Malformed input.",MiniLookupServer,301,Error,6704,7492
2011-04-29,16:28:09.481,"Malformed input.",MiniLookupServer,301,Error,6704,7492
2011-04-29,16:28:10.898,"Added value 'b' for key
'a'",MiniLookupServer,902,Verbose,6704,7492
2011-04-29,16:28:14.026,"MiniLookupServer constructor
called",MiniLookupServer,103,Information,6704,7848
2011-04-29,16:28:14.027,"New query received from client for key
a",MiniLookupServer,901,Verbose,6704,7848
2011-04-29,16:28:18.259,"MiniLookupServer service is
closed.",MiniLookupServer,903,Verbose,6704,7492
```

A Windows Event Log tartalma:



4.6.4 Teszteset

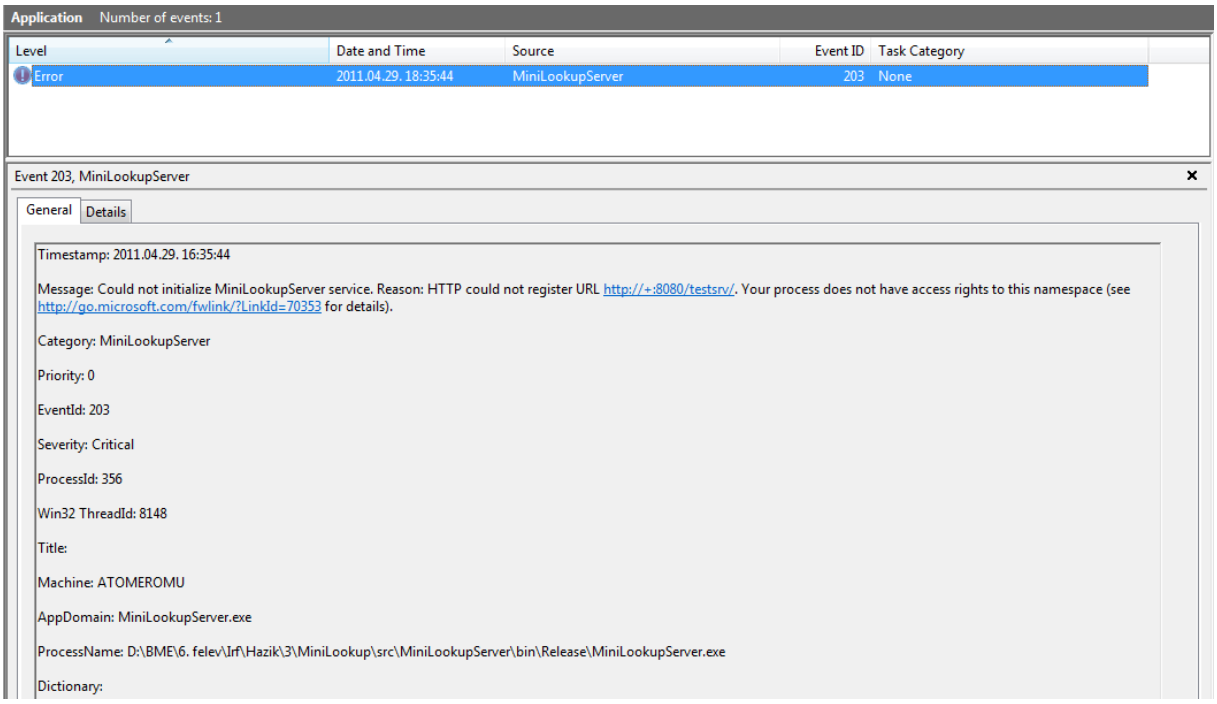
Az előző tesztesettel minden maradék eseményfajtát produkáltunk kivéve a 203-as ID-jűt. Ezt azzal váltjuk ki, hogy nem adminisztrátori módban indítjuk a szervert, de adatbázisnak megint a testsrv-t adjuk meg:

```
MiniLookupServer.exe testsrv
```

A naplófájl tartalma:

```
2011-04-29,16:35:44.529,"MiniLookupDB created for database
testsrv",MiniLookupServer,102,Information,356,8148
2011-04-29,16:35:44.615,"Could not initialize MiniLookupServer service. Reason:
HTTP could not register URL http://+:8080/testsrv/. Your process does not have
access rights to this namespace (see http://go.microsoft.com/fwlink/?LinkId=70353
for details).",MiniLookupServer,203,Critical,356,8148
```

A Windows Event Log tartalma:



4.7 A kész program ellenőrzése az FxCop és StyleCop programok segítségével

Természetesen csak a Server programot ellenőriztem, hiszen csak azon változtattam, annak átírása volt a feladat, a Client program adva volt. A StyleCop a fordítás után nem ír hibaüzenetet. A lefordított binárist FxCop-pal ellenőrizve látunk egy hibát. Én úgy ítéltem meg, hogy ez a hiba téves, az IRF névtér elnevezés esetünkben helytálló. A többi hibát kijavítottam: A MiniLookupServerhez létrehoztam egy Strong Name Key-t, illetve Neutral Language-nak English-t állítottam be.

4.8 A program telepítése

Ahhoz, hogy a forráskódot meg lehessen nyitni Visual Studio-ban a StyleCop nevű programot telepíteni kell.

Ahhoz, hogy a program a Windows Event Log-ba is tudjon naplózni regisztrálni kell a forrás nevét. Ezt a következő PowerShell paranccsal tehetjük meg:

```
[System.Diagnostics.EventLog]::CreateEventSource("MiniLookupServer","Application")
```

Fontos, hogy a powershell-t adminisztrátorként kell indítani. A MiniLookupServer.exe-t használatkor adminisztrátorként kell futtatni, a Client-et nem szükséges.

Hivatkozások

[1] BME MIT, *Hogyan készítsünk házi feladat dokumentációt és mérési jegyzőkönyvet*, elérhető online: http://www.inf.mit.bme.hu/edu/dokumentacio

[2] Erőforrások szerkesztése Visual Studio-ban: http://msdn.microsoft.com/en-us/library/7k989cfy(v=vs.80).aspx

[3] Az Enterprise Library Logging részének használatához segédlet: http://msdn.microsoft.com/en-us/library/ff953185%28PandP.50%29.aspx