Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék
Beágyazott és irányító rendszerek szakirány – Számítógép-alapú rendszerek ágazat

Rendszer és alkalmazástechnika laboratórium II.

Mérési útmutató

6. mérés

Vastagkliens alkalmazásfejlesztés PC-n I. Document-View architektúra

Laborfelelős: Szabó Zoltán (szabo.zoltan@aut.bme.hu)
A mérést kidolgozta: Benedek Zoltán (benedek.zoltan@aut.bme.hu)
Utolsó módosítás ideje: 2011.02.08 10:30

Bevezetés

Az előző félévben a Kliensalkalmazások fejlesztése tárgy keretében megismerkedtünk a vastagkliens alkalmazások fejlesztésének alapjaival. A mérés elsődleges célja, hogy az alapok gyakorlatban történő alkalmazásának képessége biztosított legyen. A mérés során egy Document-View architektúrára épülő, grafikus felhasználói felülettel rendelkező alkalmazást kell elkészíteni. A mérés a következő, a Kliensalkalmazások fejlesztése tárgy keretében tanult témakörök ismeretére épít:

- C# nyelv alapszintű ismerete
- C# property, delegate, event alkalmazástechnikája
- Windows Forms alkalmazások fejlesztésének alapjai (Form, vezérlőelemek, eseménykezelés)
- Grafikus megjelenítés Windows Forms alkalmazásokban
- Document-View architektúra elméleti ismerete és alkalmazása egyszerű környezetben
- UserControl és használata

Felkészülés a gyakorlatra

A felkészülés során – az elméleti részek átismétlése mellett - érdemes az alábbi, kliensalkalmazások fejlesztése tárgyhoz kapcsolódó gyakorlatok anyagait átismételni:

- 2. gyakorlat - Felhasználói felület kialakítása (Windows Forms, MenuStrip, ToolStrinp, Dock, Anchor, Split). Letölthető: http://www.aut.bme.hu/portal/VIAUM254 oldalon a „04-07 Előadás és gyakorlat anyagai (vastagkliens alapok)” részeként.
- 3. gyakorlat - Szoftvertervezés és grafikus megjelenítés eseményvezérelt környezetben (Breakout játék). Letölthető: http://www.aut.bme.hu/portal/VIAUM254 oldalon a „04-07 Előadás és gyakorlat anyagai (vastagkliens alapok)” részeként.
- 4. gyakorlat – Document-View architektúra alkalmazása a gyakorlatban (FontEditor). Letölthető: http://www.aut.bme.hu/portal/VIAUM254 oldalon a” 09-11 Előadás és gyakorlat anyagai (UML, architektúra)” részeként.

A mérés leírása

Feladatleírás

A mérés során egy olyan vastagkliens (Windows Forms) alkalmazást kell elkészíteni, amely képes fájlban időbélyeggel tárolt mérési értékek grafikus megjelenítésére. Az alkalmazásnak a Document-View architektúrát kell követnie. Egyszerre több dokumentum is meg lehet nyitva, illetve egy dokumentumnak több nézete is lehet. A főablak egy TabControl-t tartalmazzon, melyen minden nézet egy külön „tabfülön” jelenik meg.

Mérési feladatok

Irányelvek

A megvalósítás során használjon beszédes változóneveket, pl. pixelPerSec.

1. Feladat - a kiindulási környezet megismerése

Töltsük le a tárgy honlapjáról a kiindulási keretet. Ez egy Visual Studio solution, amely egy Document-View keretet tartalmaz. Futtatva teszteljük a kiindulási alkalmazást, majd a megjegyzésekkel ellátott forráskódot nézve ismerkedjünk meg a keret architektúrájával, működésével.

A fontosabb osztályok a következők:

- **MainForm osztály**: Az alkalmazás főablaka. Egy TabControl-t tartalmaz, ahol megjelennek az egyes dokumentumok nézetei. Kezeli a MenuStrip eseményeit, a többségük kezelőfüggvényében egyszerűen továbbhív az App osztályba.
- **App osztály**: Az alkalmazást reprezentálja. Egy példányt kell létrehozni belőle az Initialize hívásával, ez lesz a "gyökérobjektum". Ez bármely osztály számára hozzáférhető az App.Instance property-n keresztül. Tárolja a dokumentumok listáját. Legfontosabb tagjai a következők
    - documents: Valamennyi megnyitott dokumentumot tartalmazó lista.
    - activeView: Az aktív nézetet adja vissza. Ezt az aktív tabpage meghatározza, melyik az aktív view. Tabváltáskor mindig frissítésre kerül. A TabPage-ek a Tag property-jükben tárolják az a nézet objektumot, amit megjelenítenek.
    - ActiveDocument: Az aktív dokumentumot adja vissza. Ezt az aktív tabpage meghatározza, melyik az aktív view, a view meg tárolja a dokumentumot, amihez tartozik.
    - NewDocument: Létrehoz egy új dokumentumot, a hozzá tartozó nézettel. Alaposan tanulmányozzuk át az implementációt, az általa hívott függvényeket is beleértve!
    - CreateViewForActiveDocument: Egy új nézetet hoz létre az aktív dokumentumhoz. A Window/New View menüelem kiválasztásának hatására hívódik meg.
    - CloseActiveView: Bezárja az aktív nézetet.
- **Document osztály**: Az egyes dokumentum típusok ősosztálya. Bár esetünkben csak egy dokumentum típus létezik, a későbbi bővíthetőség miatt célszerű külön választani. Tartalmazza a nézetek listáját, melyek a dokumentumot megjelenítik. Az `UpdateAllViews` művelete valamennyi nézetet értesít, hogy frissítsék magukat. A LoadDocument és SaveDocument üres virtuális függvények, melyek a dokumentum betöltésekor és mentésekor

kerülnek meghívásra. A Document leszármazott osztályunkban kell felüldefiniálni és értelemszerűen megvalósítani őket.

- **IView**: Az egyes nézetek közös interfésze. Azért nem osztály, mert a nézetek tipikusan a UserControl-ból származnak le, és egy osztálynak nem lehet több ősosztálya .NET környezetben.
- **DemoView**: Egy demo implementáció nézetre. Mintaként szolgálhat saját nézet létrehozásához. A UserControl osztályból származik, és implementálja az IView interfészt.

Feladat (nem kell megvalósítani, csak egyeztesse a mérésvezetővel): az IView egy interfész, ezért a GetDocument kódját nem lehet implementálni benne. Helyette minden nézetben „copy-paste”-tel duplikálni kell a megfelelő kódot. Tudna-e elegánsabb megoldást javasolni?

2. Feladat – Mérési értékek reprezentálása, saját dokumentum osztály, adatok mentése és betöltése

2.1 Vezessen be egy osztályt a jel értékek reprezentálására

Legyen az osztály neve SignalValue, és egy Value (double) mezőben tárolja a mért értéket, az időbélyeget pedig egy TimeStamp (DateTime) mezőben.

2.2 Vezessen be egy saját dokumentum osztályt a dokumentumhoz tartozó jelértékek tárolására.

Legyen az osztály neve SignalDocument, és egy List<SignalValue>-ban (SignalValues) tárolja a jelértékeket.
Megjegyzés: az ős Document nem rendelkezik default konstruktorral, ezért kell írjuk a leszármazottunkban megfelelő konstruktort:

```
public SignalDocument(string name) : base(name)
{
}
```

Módosítsuk az App.NewDocument-et, hogy a leszármazott SignalDocument-et példányosítsa.

2.3 Gondoskodjon a dokumentum által tárolt adatok elmentéséről

- A tesztelést segítendő inicializálja a SignalDocument-ben tárolt jelérték listát úgy, hogy mindig legyen benne néhány elem (pl. a konstruktorban).
- Írja meg az App.SaveActiveDocument függvényt a megjegyzéseknek megfelelően. A SaveFileDialog használatára az MSDN Library-ben talál példát (célszerű a Google-ben rákeresni).
- Definiálja felül a SignalDocument osztályban az örökölt SaveDocument függvényt, melyben írja ki a tárolt jelértékeket, időbélyeggel együtt. A mentés során arra törekszünk, hogy tömör, mégis olvasható formátumot kapjuk. Ennek megfelelően az XML és a bináris formátum nem javasolt. Kövessük a következő szöveges formátumot:

```
10   2008-12-31T23:00:00.1110000Z
20   2008-12-31T23:00:01.8760000Z
30   2008-12-31T23:00:02.3000000Z
10   2008-12-31T23:00:03.2320000Z
```

Az első oszlopban a jelértékek, a másodikban az időpont található, asz oszlopok tabbal szeparáltak (’\t’). Az időpont legyen UTC idő, hogy ha a fájlt más időzónában töltik be, akkor is a helyes helyi időt mutassa. Az megfelelő string konverzió a következő:

```
string dt = myDateTime.ToUniversalTime().ToString("o");
```

Szöveges adatok fájlba írására a StreamWriter osztályt használjuk. Figyelem: megoldásunkban garantáljuk, hogy kivétel esetén is lezáródjon a fájlunk: használjunk try-finally blokkot, vagy alkalmazzuk a using utasítást:

```
using (StreamWriter sw = new StreamWriter(filePath))
{
    sw.Write(..);
}
```

- Az alkalmazást futtatva tesztelje a mentés funkciót.

2.4 Biztosítson lehetőséget dokumentum fájlból betöltésére

- Írja meg az App.OpenDocument függvényt a benne szereplő megjegyzéseknek megfelelően.
- Definiálja felül a SignalDocument osztályban az örökölt LoadDocument függvényt, melyben töltse fel a tárolt jelérték listát a fájl tartalma alapján. Szöveges adatok fájlból beolvasására a StreamReader osztályt használjuk, a mentéshez hasonlóan using blokkban. Ügyeljen a következőkre:
    - Az üres, vagy csak whitespace karaktereket tartalmazó sorokat át kell ugrani. A string.Trim használható a whitespace karakterek kiszűrésére, pl.:

      ```
      s = s.Trim();
      ```

    - Az oszlopok tab karakterrel szeparáltak. Egy szring adott karakter szerinti vágására kényelmesen használható a string.Split, pl.:

      ```
      string[] columns = line.Split('\t');
      ```

    - Sztringből double-t, illetve DateTime objektumot a <típusnév>.Parse(str) függvénnyel lehet pl. kinyerni:

      ```
      double d = double.Parse(strValue);
      DateTime dt = DateTime.Parse(strValue;
      ```

    - A fájlban UTC időbélyegek szerepelnek, ezt a dokumentum osztályban tárolás előtt konvertáljuk lokális időbe:

      ```
      DateTime localDt = utcDt.ToLocalTime();
      ```

2.5 A betöltést követően ellenőrizze a betöltés sikerességét

Mivel grafikus megjelenítéssel még nem rendelkezik az alkalmazás, más megoldást kell választani. Nyomkövetésre, diagnosztikára a System.Diagnostics névtér osztályai használhatók. A Trace osztály „Debug” build esetén a Write/WriteLine utasítással

kiírt adatokat trace-eli, ami alapértelmezésben azt jelenti, hogy megjeleníti a Visual Studio Output ablakában. Írjunk egy Tracevalues segédfüggvényt a SignalDocument osztályba, ami trace-eli a tárolt jeleket:

```
void TraceValues()
{
    foreach (SignalValue value in SignalValues)
        Trace.WriteLine(value.ToString());
}
```

Hívjuk meg a betöltő függvényünk végén, és ellenőrizzük a működést.

3. Feladat – Jelek grafikus megjelenítése, saját nézet osztály

3.1 Valósítsa meg a dokumentumba betöltött jelek alapszintű grafikus megjelenítését egy új a nézet osztályban.

- Az új nézet egy UserControl legyen. UserControl-t felvenni pl. a Project/Add UserControl menüvel lehet. Legyen a neve GraphicsSignalView (jelezve, hogy ez egy grafikus nézet, és nem karakteresen jeleníti meg a jeleket).
- Bővítse az osztályt a DemoView mintájára (többek között implementálja az IView interfészt)
- Módosítsa az App.createView()-t, hogy DemoView helyett GraphicsSignalView-t hozzon létre.

*3.2 Az OnPaint-t override-olva valósítsa meg a jelek kirajzolását. Először 3*3 pixeles „pontokat" rajzoljon (pl. Graphics.FillRectangle-lel), majd a pontokat kösse össze vonalakkal.*

A megvalósításban segíthet a következő:

- Két DateTime érték különbsége egy TimeSpan típusú objektumot eredményez.
- Egy DateTime objektum a Ticks property-jében adja vissza legjobb felbontással az általa tárolt időértéket (100 nsec felbontás).

3.3 Biztosítson lehetőséget a nézet nagyításra és kicsinyítésére. Ehhez helyezzen el egy kisméretű, + és – szöveget tartalmazó nyomógombot a nézeten.

3.4 Biztosítson lehetőséget a grafikon görgetésére

A megvalósításban használhatunk egyedi scrollbar-t is, de ennél egyszerűbb és elegánsabb a UserControl autoscroll támogatását felhasználni. Ehhez először engedélyezzük az AutoScroll-t a properties ablakban a nézethez tartozó UserControl-ra. Ezt követően meg kell adni a rajzolófelület nagyságát, aminél ha kisebb a UserControl-unk mérete, akkor automatikusan megjelenik a megfelelő scrollbar:

```
this.AutoScrollMinSize = new Size(widthInPixels, heightInPixels);
```

Ezt követően a kirajzolás során a rajzot az aktuális scroll pozíciónak megfelelően el kell tolni. Erre a legegyszerűbb megoldás, ha egy, a scroll pozíciónak megfelelő eltolást eredményező transzformációs mátrixot állítunk be a Graphics objektumra a kirajzolás előtt:

```
Matrix transform = new Matrix(1, 0, 0, 1, AutoScrollPosition.X,
    AutoScrollPosition.Y);
g.Transform = transform;

… rajzolás …
```

Egyeztesse a mérésvezetővel, hogy lehet-e ennél hatékonyabb megoldást alkalmazni!

4. Feladat – Élő jelmegjelenítés

4.1 Biztosítson lehetőséget élő jelmegjelenítésre, a következőknek megfelelően:

- Támogatni kell ez addigi, fájl alapú megjelenítést.
- A felhasználó a fájl alapú megjelenítés mellett élő üzemmódban is jeleníthet meg jeleket. Egy jel megjelenítését kell csak így támogatni, de akár több nézetben is. Az élő megjelenítés a fájlalapúhoz hasonlóan külön tabokon jelenjen meg.
- Az élő megjelenítés lényege, hogy új mérések a megjelenítés közben is születnek. A valóságban pl. hálózaton, soros porton, vagy valamely perifériáról, esetünkben azonban legyenek ezek mesterségesen generáltak.

Segítség a megoldáshoz:

- A mérések mesterséges generálását Timer-el oldjuk meg (System.Windows.Forms.Timer). Az App osztályunkban vegyünk fel egy Timer objektumot, StartGenerate és egy StopGenerate függvényt. A StartGenerate elindítja a timert, a StopGenerate leállítja, a timer eseménykezelő pedig meghívja a megfelelő dokumentum AddMeasuredValue függvényét. Az AddMeasuredValue felveszi a paraméterben megkapott mérési objektumot a mérési lista végére.

5. Feladat (extra) – Zoom a toolbar-ról combóbox segítségével

4.1 Biztosítson lehetőséget az aktív nézet nagyítására/kicsinyítésére a toolbar-ról. A felszhasználó egy combóbox-ból választhasson előredefiniált értékek közül, illetve adhasson meg egyedi értéket is.

Utólagos otthoni gyakorlási lehetőség

Utólagos, otthoni gyakorlásra alkalmas a következő feladat. Jelen megoldásunkban egy adatfájl és dokumentum csak egy mérési adatsorozatot tartalmazhat. A megoldás általánosítható olyan módon, hogy egy adatfájl és egy dokumentum is több mérési adatsorozatot tartalmazzon. Ekkor egy nézet egy adatsorozatot jelenít meg.

Ellenőrző kérdések

- Ismertesse a C# delegate és event fogalmát, mutasson példát a használatára!

- Ismertesse a Windows Forms alkalmazások architektúráját!
- Ismertesse a grafikus megjelenítés mechanizmusát (Paint, érvénytelen terület, Graphics osztály, stb.).
    - Gyakorlásképpen készítsen egy olyan alkalmazást, ami a főablak közepén másodpercenként egyel növeli egy számláló értékét!
- Ismertesse a Document-View architektúrát, az architektúrában szereplő osztályok szerepét, az osztályok kapcsolatát! Írjon C# nyelven egy olyan egyszerű Document és View osztály, amely rendelkezik a szükséges tagváltozókkal és tagfüggvényekkel, és ami megvalósítja a szükséges működést (pl. nézetek frissítése)!
- Ismertesse a UserControl szerepét!