Integration mellan två gamla system

Mikael Johannesson, 2022-03-10, SoftHouse

Uppgift från kund

Det ena systemet levererar ett radbaserat filformat medan det andra kräver XML. Du ska skriva en konverterare som bygger upp rätt XML-struktur.

Indata

Filformat:

P|förnamn|efternamn
T|mobilnummer|fastnätsnummer
A|gata|stad|postnummer
F|namn|födelseår
P kan följas av T, A och F
F kan följas av T och A

Exempel:

P|Elof|Sundin
T|073-101801|018-101801
A|S:t Johannesgatan 16|Uppsala|75330
F|Hans|1967
A|Frodegatan 13B|Uppsala|75325
F|Anna|1969
T|073-101802|08-101802
P|Boris|Johnson
A|10 Downing Street|London

Utdata

Ger XML som:

```
<people>
<person>
<firstname>Elof</firstname>
<lastname>Sundin</lastname>
<address>
<street>S:t Johannesgatan 16</street>
<city>Uppsala</city>
<zip>75330</zip>
</address>
<phone>
<mobile>073-101801</mobile>
<landline>018-101801</landline>
</phone>
<family>
<name>Hans</name>
<born>1967</born>
<address>...</address>
</family>
<family>...</family>
</person>
<person>...</person>

</people>
```

Angrepp

Analys och teknival

- Stoppade in xml-koden i en xsd-generator online, se fil ‘Schema.xsd’ för att säkerställa att jag inte missat något.
- *kan det finnas 0..1 av och de kan vara inkompletta, förutsätter i så fall att det är den sista som saknas, alltså att om bara ett attribut finns så är det .*
- kan det finnas 0…1 av, dito ovanstående att de kan vara inkompletta, förutsätter i så fall att det är den första i listan som finns om det bara finns en.
- Jag tänker mig relativt små datamängder, så små att all data i indata får plats att bearbetas i datorns internminne.
- Jag tänker mig att eftersom allt får plats i internminnet i mitt tankeexperiment så gör jag så att jag bygger en lista av person-objekt och när den är klar så kan jag skriva ut xml-koden för hela alltet från metoder i dessa objekt.

- En tanke med att ha en lista av objekt som mellanled är att man i en verklig situation då skulle kunna presentera data, flagga upp tveksamheter för operatören i samband med konverteringen.
- Väljer att inte jobba med filstrukturer, indata- och utdata hamnar därmed i samma mapp som exe-filen ligger i vilket är ok i denna demonstration.
- Förenklar och sätter konstanter i koden för filnamn och mappstruktur för in- och utdata, default är de båda filerna indata.txt och people.xml . I en verklig lösning bör man naturligtvis kunna sätta sådan här data i gränssnittet, tänker att detta är en rimlig förenkling för en demonstration dock!
- Jag har mycket rudimentär felhantering, hanterar att data saknas på ett antal ställen men inte mycket mer.
- Väljer att inte jobba med klassen XML vilket kanske är mer udda, tar tid att sätta in sig i och jag ser främsta fördelen med denna klass att den fungerar även när all data inte får plats i internminnet utan att man skriver löpande till fil, det behövs inte för mig, bedömer att jag då får bättre överblick om jag själv bygger xml-koden i person, avatar och familj-klasserna.
- En mindre test av XML-klassen har gjorts, visade sig vara en utmaning att få snygg kod om man som jag mellanlandar i en lista av objekt innan man skriver ut.
- Ett UML-diagram över klasser har tagits fram, finns som uml_objects.vsd (samt uml_objects.pdf)
- Ett flöde över inläsning från PTAF till lista av objekt har tagits fram och återfinns i källkoden Program.cs
- Inget krav på teckentabeller så vi förutsätter att samma teckentabell används i både in- och utdata. Bör stämmas av mot fler exempel med riktig data!
- Inga krav på grafisk presentation så jag väljer en console-app.
- Språk C#
- För att snabba upp väljer jag att använda den (gamla) version av Visual Studio som råkar finnas på min kommunikationsdator, VS2012. Jag hoppas att ni fortfarande kan köra exe-filen utan att behöva installera några bibliotek för äldre .net.

Flöde

Hämtat från koden

```
/* Flöde
 *
 * vi förutsätter tills vidare att indata är korrekt, ingen kod för felhant
 * vi förutsätter att listorna är av rimlig storlek och gott och väl får pl
 * vi jobbar initialt med att testdata ligger i Program.cs som en array av
```

```
* vi tänker oss att utdata, xml-varianten, skrivs till Console tills vidar
*
* vi väljer att skriva vår egen xml-parser trots att vi vet att det finns
* detta då den främsta fördelen med att använda klassen är att man jobbar
* förlorar över den komplexitet som måste införas (många rader kod i onöda
* för mig med liten skärm... kanske växlar vi sedan när denna modell funge
*
* läs indata och lägg i lista av objekt
*
* bool pExist = false
* bool fExist = false
* skapa p-objekt (för att få igenom koden)
* skapa f-objekt (för att få igenom koden)
*
* för varje rad:
    * Om P
    *  om fExist, skriv f till f-lista, skriv p till p-lista
    *  om pExist, skriv p till p-lista
    *  skapa nytt p
    *  sätt pExist, !fExist
    *  populera p med förnamn, efternamn
    *
    * Om F
    *  om fExist, skriv f till f-lista
    *  skapa nytt f
    *  sätt fExist
    *  populera f med name, dob
    *
    * Om A
    *  om fExist populera aktuellt f med A
    *  annars (!fExist), populera aktuellt p med A
    *
    * Om T
    *  om fExist, populera aktullt f med T
    *  annars (!fExist), populera aktuellt p med T
*  om fExist, skriv f till f-lista och p till p-lista
*  om !fExist men pExist (minst en post), skriv p till p-lista
*
*
* för varje p i p-lista:
   *  skriv xml(namn)
   *  om T angivet, skriv xml(T)
*  *  om A angivet, skriv xml(A)
*  *  för varje f i f-lista: (kanske finns inga men då kör vi noll varv)
*        * skriv xml(namn, dob)
*        * om T angivet, skriv xml(T)
```

```
        *         * om A angivet, skriv xml(A)
        *
        *
        *
        */
```