Tallinna Ülikool
Informaatika Instituut

Lauri Mattus

Microsoft Dynamics AX arendamine

Seminaritöö

Juhendaja: Jaagup Kippar

Autor: ....."....." ....2008. a.
Juhendaja: ....."....." ..... 2008. a.
Instituudi direktor: ....."....." ..... 2008. a.

Tallinn 2008

Sissejuhatus

Microsoft Dynamics AX on suurettevõtte ressursiplaneerimise tarkvara, mida arendati esialgu nime all Axapta. Axapta arendus sai alguse 1983 aastal Taani firmas Damgaard Data A/S ning esimene versioon valmis aastal 1998. Kaks aastat hiljem ühines Damgaard teise Taani firmaga Navision, mis tegeles samuti majandustarkvara arendamisega. Aastal 2002 ostis Microsoft ühinenud firma Damgaard Navision'i ning Navison Damgraad Axapta sai nimeks Microsoft Business Solution Axapta. Aastast 2008 kannab Axapta nime Microsoft Dynamics AX.[1]

Käesoleval hetkel on uusim versioon Microsoft Dynamics AX 4.0 ning uus versioon tuleb välja aastal 2009 nimega Microsoft Dynamics AX 2009.

Antud seminaritöö on koostatud eesmärgiga tutvustada Microsoft Dynamics AX arendamist versiooni 4.0 baasil. Vaatluse alla on võetud just nimelt arendusmeetodid ja –vahendid, mitte ressursiplaneerimises toimivad protsessid. Eeldatud lugeja mõningased teadmised programmeerimisest ja mõne kasutatavama programmeerimiskeele tundmine (soovitavalt c, c++, c# või java). Microsoft Dynamics AX on väga võimas ja multifunktsionaalne tarkvara, sama kehtib arendusmeetodite ja –võimaluste kohta. Põhieesmärgiks antud seminaritööl on luua ettekujutus Microsoft Dynamics AX põhilisetest komponentidest, millega puutub arendaja kokku oma igapäevatöös.

Töö on jaotatud kuute suuremasse peatükki. Esimeses peatükis on ülevaade arendusvõimalusest ja tarkvara arhitektuurist. Teine peatükk tutvustab rakendusobjektide puud, mis on iga arenduse keskmeks. Kolmas peatükk käsitleb Microsoft Dynamics AX'is kasutatavat programmeerimiskeelt X++. Viimased kolm peatükki kirjeldavad enimkasutavaid komponente ning nende arendamist.

Sarnasel teemal on Tallinna Ülikoolis üks bakalaureusetöö loodud [6], kuid siis sai keskendatud pigem konkreetsele logistikarakendusele. Siinses töös vaadeldakse aga Microsoft Dynamics AX ning X++ tehnilisi võimalusi laiemalt.

Sisukord

1. Microsoft Dynamics AX lühitutvustus

Microsoft Dynamics AX on ettevõtte ressursiplaneerimise tarkvara, mis toetab mitmeid keeli ja erinevaid valuutakursse. Microsoft Dynamics AX sobib hästi tootmise, e-kaubanduse, hulgimüügi ja teenuste pakkumisega tegelevatele firmadele. See on täielikult integreeritav lahendus koos veebitoega ning toetab Microsoft SQL Serverit ja Oracle'it. Samuti omab see tarkvara kohandatavat baaskoodi, mida saab muuta vastavalt kliendi vajadustele. [2, 2 lk 2]

1.1 Microsoft Dynamics AX omadused ja võimalused

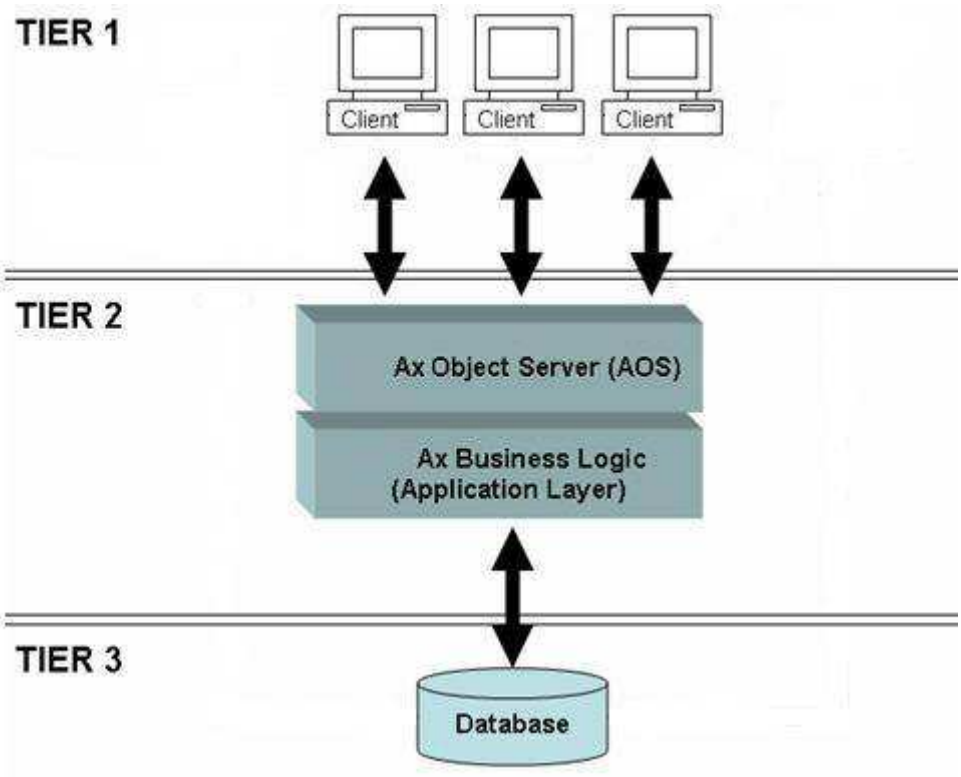
Microsoft Dynamics AX poolt pakutavad võimalused ning omadused võib jagada kahte ossa:

- Funktsionaalsed omadused ja võimalused:
  - Kõik organisatiooni kuuluvad ettevõtted kasutavad ühte ja sama andmebaasi.
  - Kõik funktsionaalsed valdkonnad nagu ressursiplaneerimine, müük ja tootmine on omavahel tihedalt integreeritud.
  - Dimensioonipõhine süsteem tootmis- ning finantsmoodulitele.
  - Võimalus ennustada ja planeerida ressursse põhinedes statistikale.
  - Mitme valuuta ning mitme keele võimalused.
- Arenduskeskkonna omadused ja võimalused:
  - Dynamics AX MorphX on integreeritud arenduskeskond (IDE) arendajatele.
  - .NET Business Connector pakub ligipääsu tervele Microsoft Dynamics AX tarkvaraliidesele lubades seda kergelt integreerida kolmanda osapoole tarkvaraga ja veebilahendustega.
  - Application Object Tree on selgroog, mis ühendab endas tarkvara kõiki muudetavaid komponente.
  - Drag-and-Drop funktsionaalsus.
  - Projektipõhine organiseeritus, et organiseerida ja jälgida muudetud komponente.[3, lk 2]

1.2 Arhitektuur

Dynamics AX omab kolmekihilist struktuuri (*3-tier*):

- Esimene kiht – klient: tegeleb kasutajaliidse ning vajaliku programmiloogikaga.
- Teine kiht – AOS (Application Object Server): tegeleb äriloogikaga
- Kolmas kiht – andmebaasi server.



Joonis1. Kihtide suhtlus diagramm [3, 3]

Firma suuruse kasvades suureneb ka kasutajate hulk. Selleks puhuks on võimalik lisada teisele kihile lisa AOS. Nii jaotub koormus kahele serverile, mis kindlustab keskkonna kiire toimimise.[3, 4]

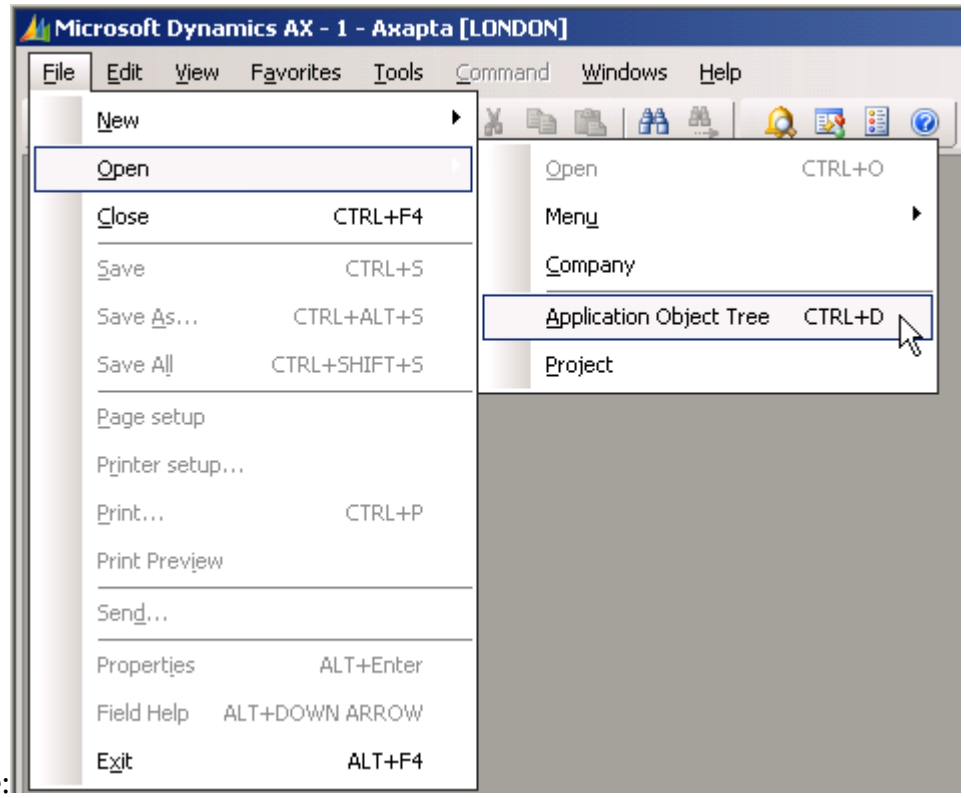
2. Rakendusobjektide puu

Rakendusobjektide puu (Application Object Tree – edaspidi AOT) on puu kujuline vaade kõikidest rakenduse objektidest (application objects). Selles paikneb kõik vajalik Dynamics AX välimuse ning funktsionaalsuse kohandamiseks. Võimalus on kasutada Drag-and-Drop'i, et luua ja muuta application objekte ilma koodi kirjutamata. [3 lk 54]

AOT on üks põhivahendeid, millega puutub kokku Microsoft Dynamics AX arendaja kokku oma igapäevatöös.

AOT'le ligipääsemiseks on 3 võimalust:

1. Avades File menüü-> Open -> Application Object Tree:

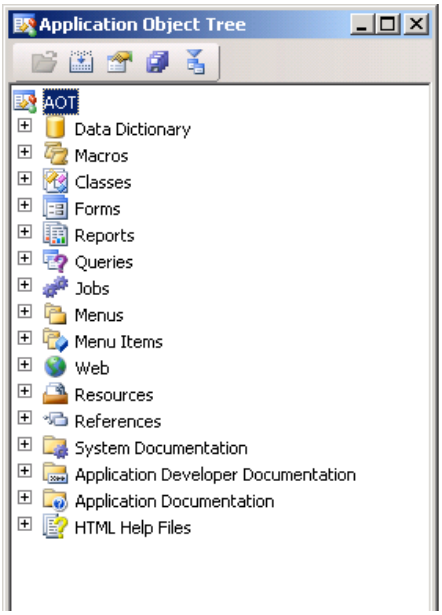


Joonis 2. AOT avamine File menüüst [3 lk 55]

2. Klikkides AOT ikoonile:

Jooni 3. AOT avamine ikoonilt [3 lk 55]

3. Vajutades CTRL + D:



Joonis 4. AOT [3 lk 54]

2.1 Data Dictionary

Joonis 5. Data Dictionary alajaotis AOT's

Antud grupist vaatleks järgmiseid enimkasutatavaid valikuid:

- *Tables* – Siit kaudu pääseb ligi andmebaasi tabelitele. Saab lisada, muuta, kustutada ja vaadata nii tabeleid kui nendes paiknevaid andmeid. Samuti saab lisada andmebaasidele meetodeid.
- *Extended Data Types* – Laiendatud andmetüübid. Siit saab lisada ja muuta laiendatud admetüüpe.
- *Base Enums* – Valik tüüpi muutujad, mille aluseks on täisarvulised muutujad.

3. Programmeerimiskeel X++
3.1 X++ Ülevaade

X++ on object-orjenteeritud programmeerimiskeel midas kasutatakse Microsoft Dynamics AX arendamisel. Tooks välja mõned omadused:

- Tüübikindlus (type safe)
- Võimaldab pärineda klassidel ainult ühest klassist (single inheritance)
- X++ ei ole tõstutundlik (case insensitive). Näiteks Name ja name ning NAME käsitletakse sama muutujanimena.
- Polümorfism
- Võimalus luua abstraktseid klasse (abstract class) ja liideseid (interface)
- Ülekirjutamine (overriding)

Programmeerijad pääsevad X++ abil ligi Microsoft Dynamics AX süsteemi klassidele, mis pakuvad funktsionaalsust alates tavalisest Input/Output, XML'ist kuni graafilise kasutajaliidese muutmiseni reaalajas. Samuti saab üles ehitada uusi klasse, mis pärinevad nendest süsteemi klassidest.[4 lk 9]

X++ oma süntaksilt on sarnane C++'le ja Java'le. Üritan anda ülevaate sellest, mis on antud keeles omapärast

3.2 Andmetüübid
3.2.1 Lihttüübid

Lihttüübid (simple data types) keeles X++ on:

Andmetüüp	**Kirjeldus**	**Näide**	**Deklaratsioon**
String	Sümbolite jada. On olemas vasakjoondusega (left aligned), paremjoondusega(right aligned), määratud pikkusega, määramata pikkusega	Nimi	str
Integer	Täisarv	2000	int
Real	Ujukomaarv	3.14	real
Date	Sisaldab päeva, kuud ja aastat	24\11\08	date
Enum	Sisemiselt käsitletakse kui täisarve. Valik, kus esimene on int 0, teine int 1 jne.	NoYes	peab deklareerima kõigepealt Base Enum'ina Data Object Tree's
Boolean	Saab omada väärtust tõene/väär (true and false)	true	boolean

Tabel 1. X++ lihttüübid [4 lk 46]

3.2.2 Liittüübid

Liittüübid (composite data types) keeles X++ on:

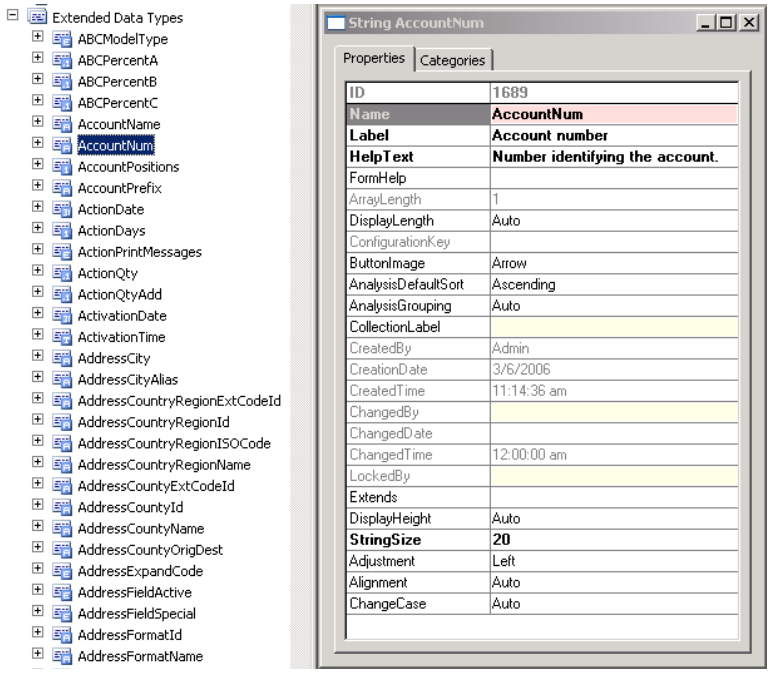
Andmetüüp	**Kirjeldus**
Array	Massiiv ühte tüüpi elementidest
Container	Dünaamiline nimistu elementidest. Võib sisaldada eri tüüpi elemente, samuti liittüübist elemente
Class	Klass on tüübidefinitsioon, mis kirjeldab ära nii muutujad kui meetodid, mis on olemas klassist tehtud objektil.
Table	Kõik tabelid, mis on defineeritud andmebaasis (data directory'is) saab käsitleda kui klasse.

Tabel 2. X++ Liittüübid [4 lk 48]

3.2.3 Laiendatud andmetüübid

Laiendatud andmetüüpe (extended data types - EDT) saab lisada ainult AOT's. Nende aluseks saavad olla kas lihttüübid (simple data types) või teised laiendatud andmetüübid.

Näide:



Joonis 6. Näide AccountNum properties

Siit on näha, millised parameetrid saab ühel laiendatud andmetüübil määrata. Näiteks väljas *StringSize* saab määrata antud andmetüübi pikkuse.

Laiendatud andmetüüpide kasulikkuse seisneb selles, et nendest pärinevatel (inheritance) andmetüüpidel säilivad kõik vanema (parent) omadused.

Microsoft Dynamics AX'is on eeldeklareeritud suur hulk erinevaid laiendatud andmetüüpe. Näiteks eraldi andmetüübid on *SalesPrice*, *ItemPrice* jne. Mis kõik pärinevad ühest tüübist *Price*, mis omakorda pärineb lihttüübist *real*.

3.3 Muutujate deklareerimine

Lihttüübi deklareerimise süntaks on järgmine:

```
andmetüüp muutujanimi;
```

Mõned näited:

```
// Integer
int a = 1;            // Deklareerimine ja initsialiseerimine teostatakse samaaegselt
int b;                // Deklareerimine ja
b = 30;               // initsialiseerimine teostatakse kahes järgus

// String
str name;             // Tavaline string tüüpi muutuja, ilma pikkuse ja joonduseta
str 10 name;          // String tüüpi muutuja, mille maksimaalne pikkus on 10
str 10 right name; // String tüüpi muutuja, mille maksimaalne pikkus on 10 ja joondus on paremale

// Date
date d;
d = 24/12/2004;   // Pane tähele, kuupäeva ümber pole “ ega ‘ ja eraldajaks on / (24 november 1959)

// Boolean
boolean b;          // Deklareerimisel saab boolean väärtuseks false
b =  true;          // siin omistatakse väärtus true

// Enum
SalesStatus status;                  // Enum tüüpi muutujale saab omistada eeldeklareeritud valikuid
status = SalesStatus::Delivered;  // antud SalesStatus enum’il on 5 võimalikku väärtust, millest üks on
// Array                             // Delivered (saadetud).
 str 20 txt[];       // Massiiv string’idest, millel maksimumpikkus 20
str 20 txt[10];  // Massiiv string’idest, millel maksimumpikkus 20 ja maksimaalne elementide arv 10

txt[0] = ‘Kalle’;  // Omistame massiivi txt esimeseks väärtuseks ‘Kalle’
txt[1] = ‘Malle’; // Omistame massiivi txt teiseks väärtuseks ‘Malle’
```

Joonis 7. Lihttüüpide deklareerimine

3.3.1 Konteinerid

Konteinerid (containers) saavad sisaldada eri tüüpi muutujaid, massiive, teisi konteinereid ning kõiki lihttüüpe, liittüüpe ja laiendatud tüüpe. Ainus, mida konteiner ei saa sisaldada on klassid. [4 lk 50]

Konteineri muutujate manipuleerimiseks on mitmeid meetodeid: conPeek(), conDel(), conNull(), conFind(), conIns() [4 lk 50]

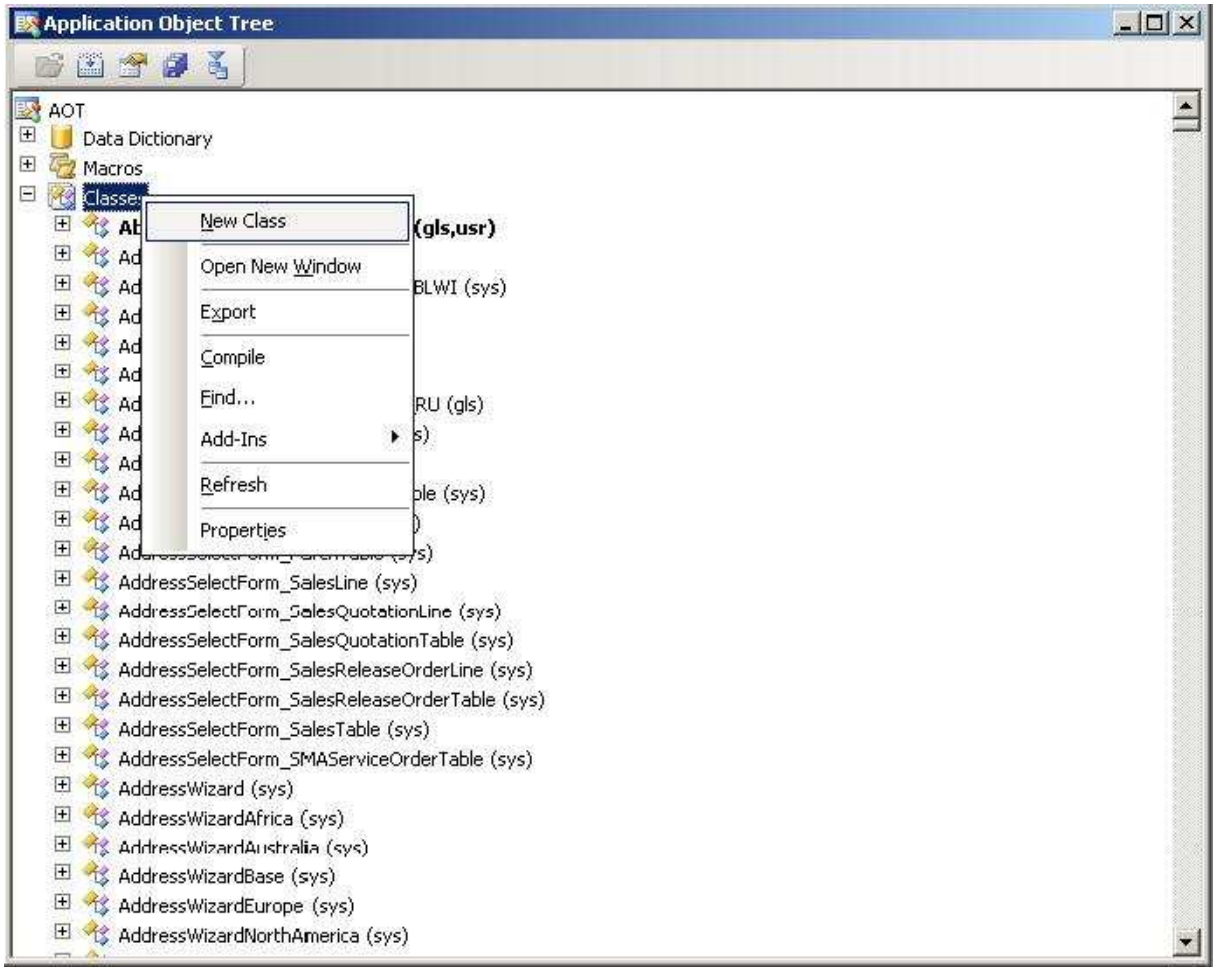
Näited:

```
container c = [10, 20, “test”,  ABC::A];

conPeek(c, 3); // tagastab konteinerist c väärtuse, mis on kohal 3 (lugema hakatakse numbrist 1)
                    // antud juhul tagastab stringi “test”

container c;            // deklareeritakse konteiner c
str t1, t2, t3;          // deklareeritakse kolm string muutujat t1, t2, t3
c = [‘a’, ‘b’, ‘c’];    // antakse väärtused konteineri muutujatele
[t1, t2, t3] = c;       // omistatakse konteineri muutujate väärtused string muutujatele t1,t2, t3
```

Joonis 8. Konteineri deklareerimine

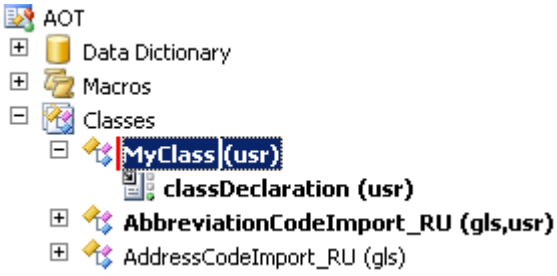
3.3.2 Klassid

Samuti nagu laiendatud andmetüüpe, saab ka klasse luua AOT'st. Selleks tuleb klikkida hiire parema klahviga jaotisel *Classes* ning valida *New Class:*



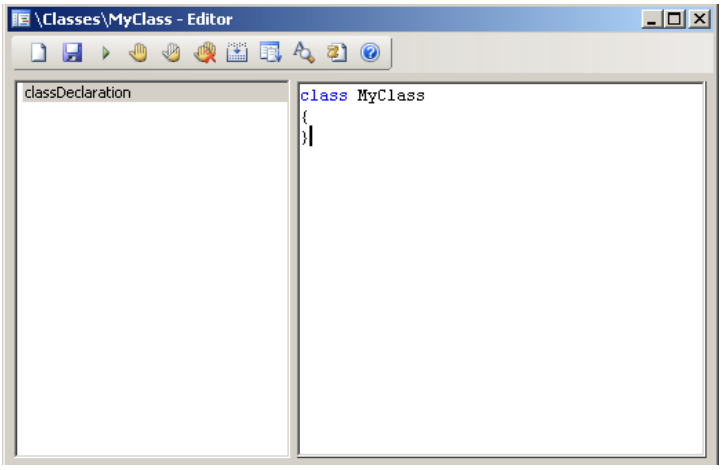
Joonis 9. Uue klassi loomine

Loome klassi nimega *MyClass*:



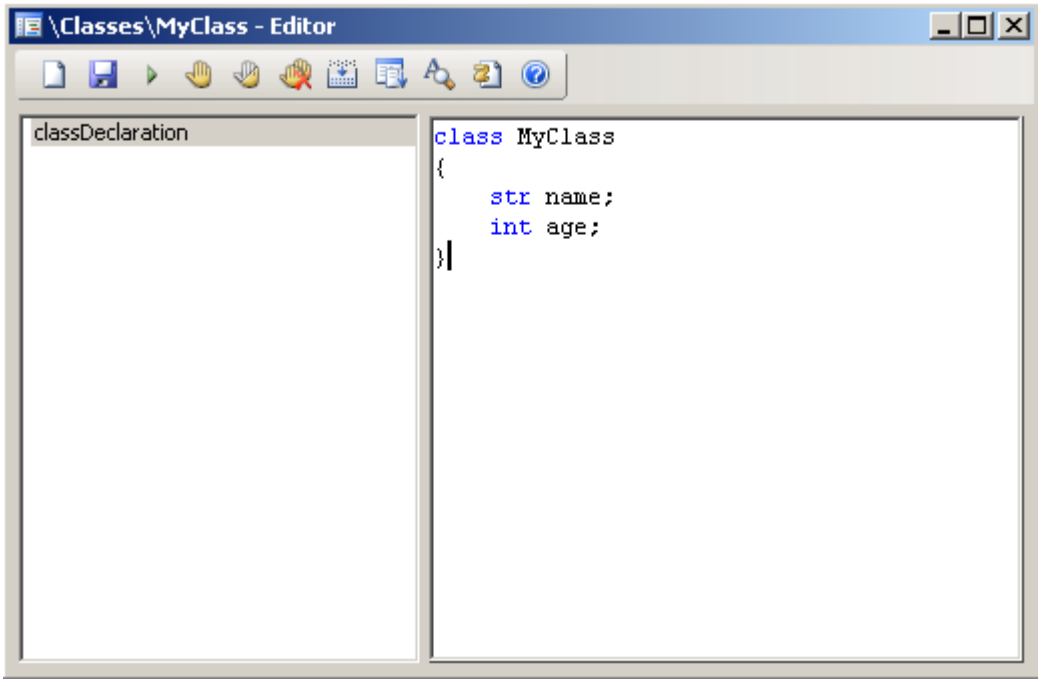
Joonis 10. Vaatepilt pärast uue klassi loomist

See meetod on teistest meetoditest erinev. See pole nagu päris meetod vaid tegelikult koht, kus deklareeritakse klassi muutujad, päritavus ning liidesed. Klikates sellele meetodile hiire vasaku klahviga 2 korda, avaneb meile redaktor:



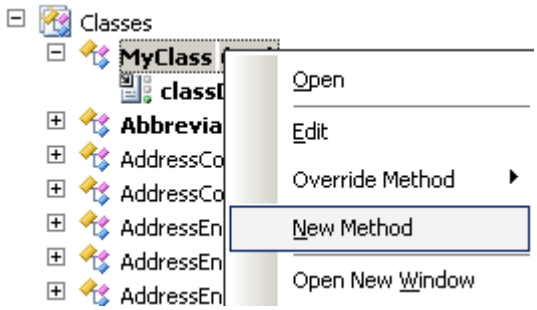
Joonis 11. Klassi meetodod classDeclaration sisu

Deklareerime klassile kaks muutujat: *name* (nimi) ja *age* (vanus).



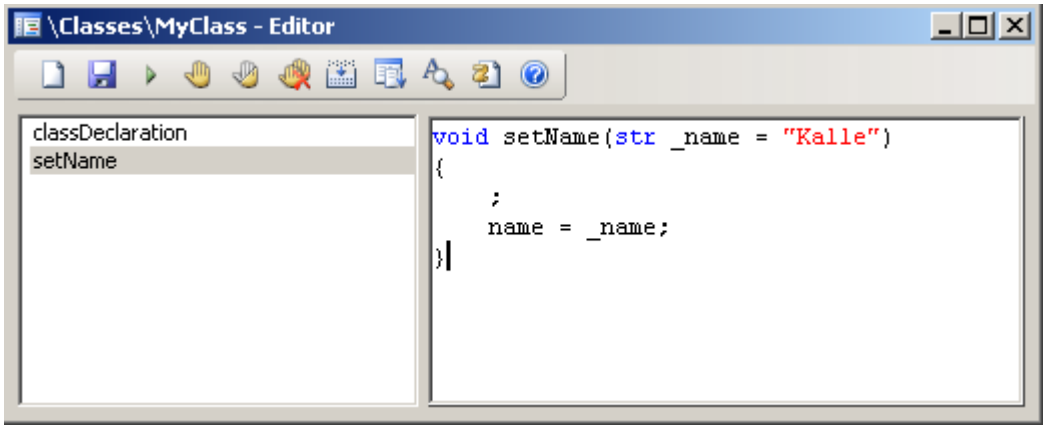
Joonis 12. Klassi muutujate deklareerimine

Järgmiseks defineerime funktsioonid: *setName()* ning *getName()*, vastavalt parameetri seadistamiseks ja küsimiseks. Selleks klikime hiire parema klahviga klassil ning valime *new method:*



Joonis 13. Uue meetodie loomine

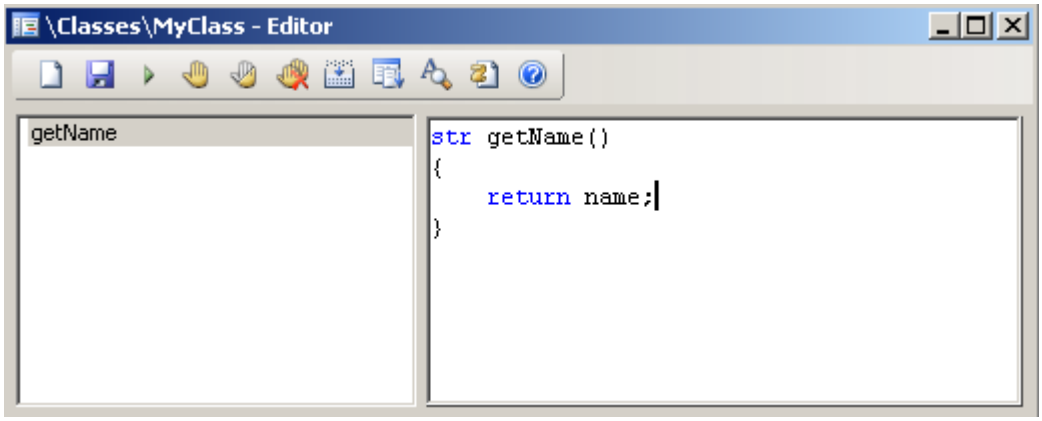
Avaneb uus redaktori aken, kuhu kirjutame:



Joonis 14. Meetodi setName() defineerimine

X++ keeles saab meetodi parameetrite väärtused vaikeväärtustada. See tähendab, et kui sellele meetodile ei anta kaasa parameetrit, siis väärtustab ta parameetri *name* automaatselt Kalle”ks.

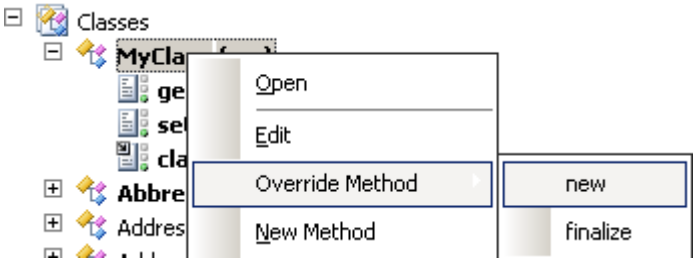
Teeme ka teise meetodi:



Joonis 14. Meetodi setName() defineerimine

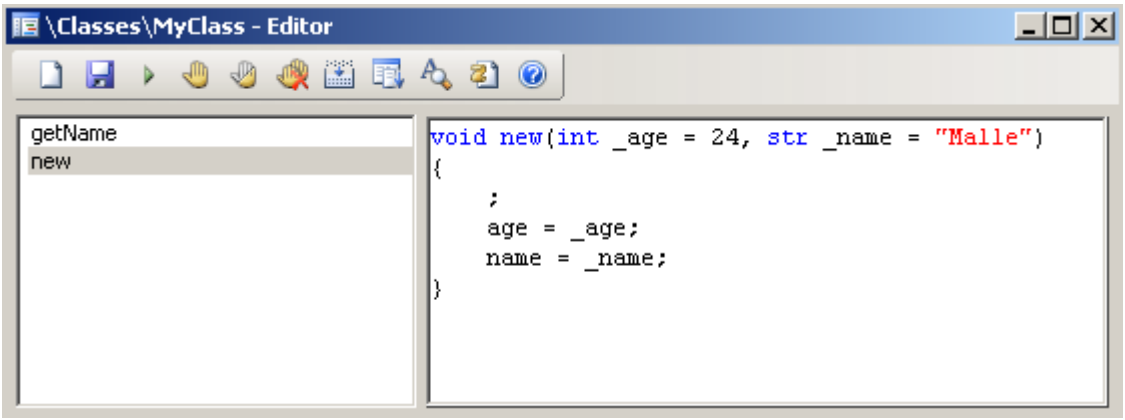
Analoogiliselt tuleks teha ka meetodid *setAge()* ning *getAge()*.

Järgmiseks aga teeme klassile konstruktori. Peidetult on igal klassil konstuktor olemas ning kui me ei soovi objekti loomisel mingeid muutujaid algväärtustada ega mingit muud funktsionaalsust lisada võib konstuktori jätta tegemata. Seekord teeme siiski konstruktori koos algväärtustamisega. Selleks klikime klassil parema hiire klahviga ning valime *Override Method -> new*. Nagu näha on siin kaks valikut: *new* käivitatakse objekti loomisel (konstruktor) ning *finalize* objekti hävitamisel (destruktor).



Joonis 15. Meetodi ülekirjutamine

Avaneb jälle redaktori aken, mille täidame järgmiselt:



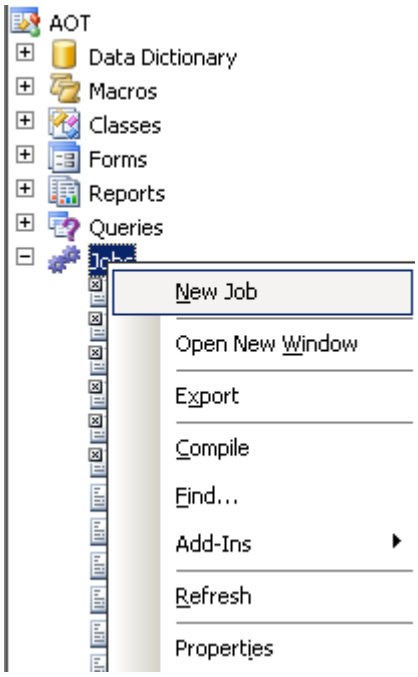
Joonis 16. Meetodi new redaktor

Nagu näha antud näitest, ei ole X++'is vaja teha erinevaid konstruktoreid vastavalt sellele, mis parameetreid konstruktorile tahetakse ette anda. Kui tekitame objekti ilma ühtegi parameetrit kaasa andmata, siis algväärtustatakse mõlemad muutujad. Kui anname kaasa ühe parameetri, siis algväärtustatakse teine. Kui anname kaasa mõlemad parameetrid, siis saavad muutujad väärtused nendest parameetritest. Nüüd on lihtne klass valmis, järgmiseks tuleks seda testida. Lihtsaim koht, kust koodi Microsoft Dynamics AX'is käivitada on "tööd "(Jobs).

4. Tööd

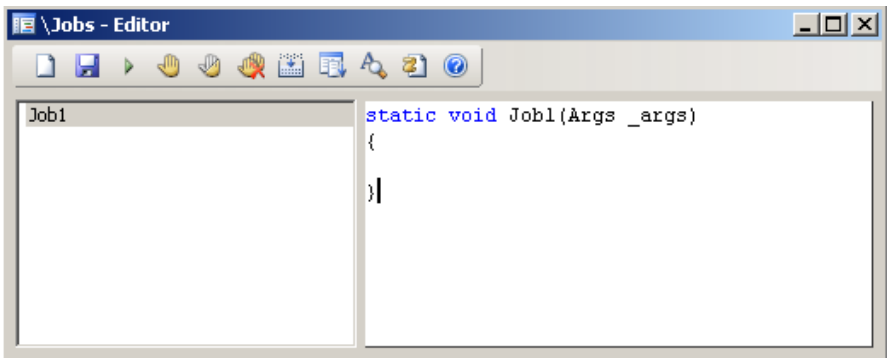
Töö (job) on tükk koodi, mille Microsoft Dynamics AX käivitab järjestikku (sequentially). Peamiselt kasutatakse töid selleks, et testida mingi koodijupi toimimist või selleks, et importida mingeid suuri koguseid andmeid andmebaasis. Samuti selleks, et käivitada mingeid ühekordseid protsesse. [4 lk 23]

Töid saab lisada AOT'st. Selleks valime jaotise *Job*, klikime sellel hiire parema klahviga ning valime *New Job*:



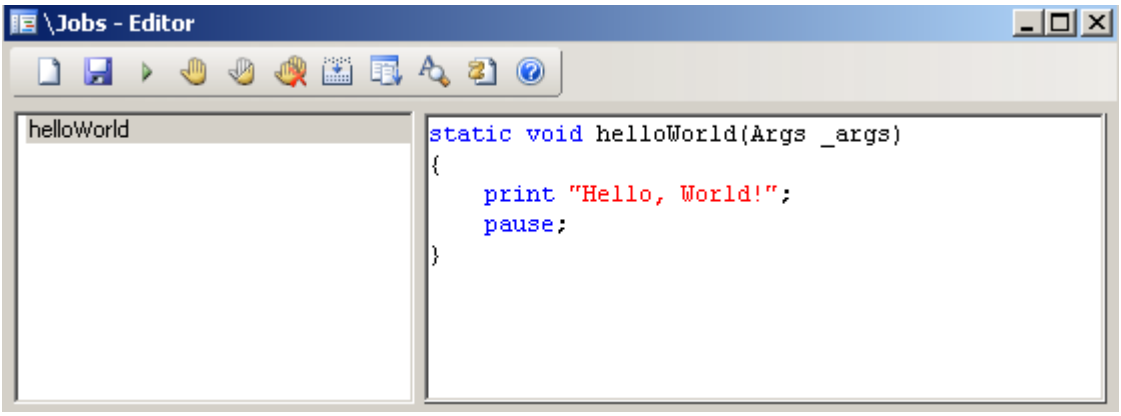
Joonis 17. Uue töö lisamine

Avaneb juba tuttavaks saanud redaktoriaken.



Joonis 18. Uus töö redaktoris

Alustame, nagu iga programmeerimiskeele õppimistki, programmiga *"Hello, World!"*. Selleks täidame meetodi sisu järgmiselt:

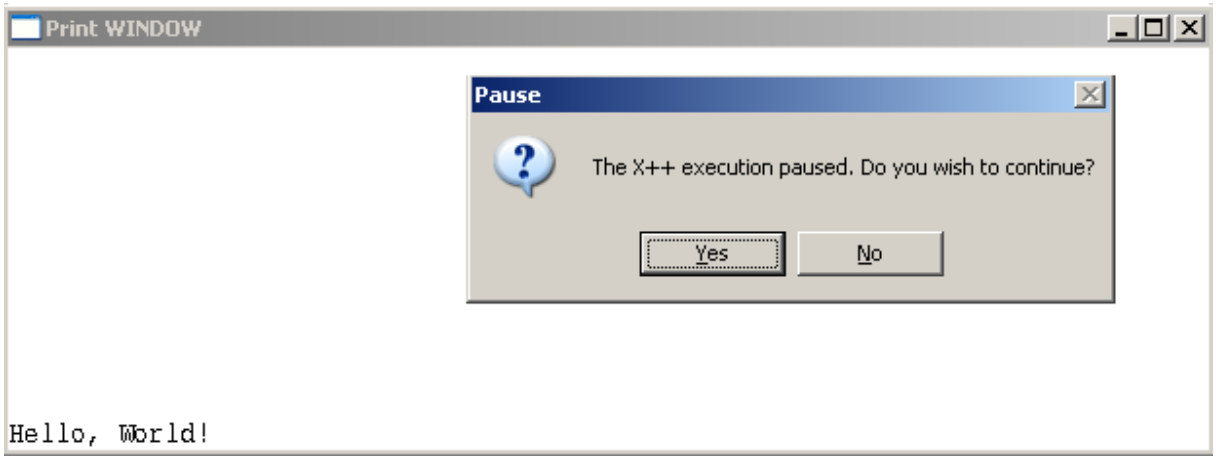


Joonis 19. Hello, World! programmijupp

print on meetod, mis avab teateakna ning prindib sinna soovitud teksti.

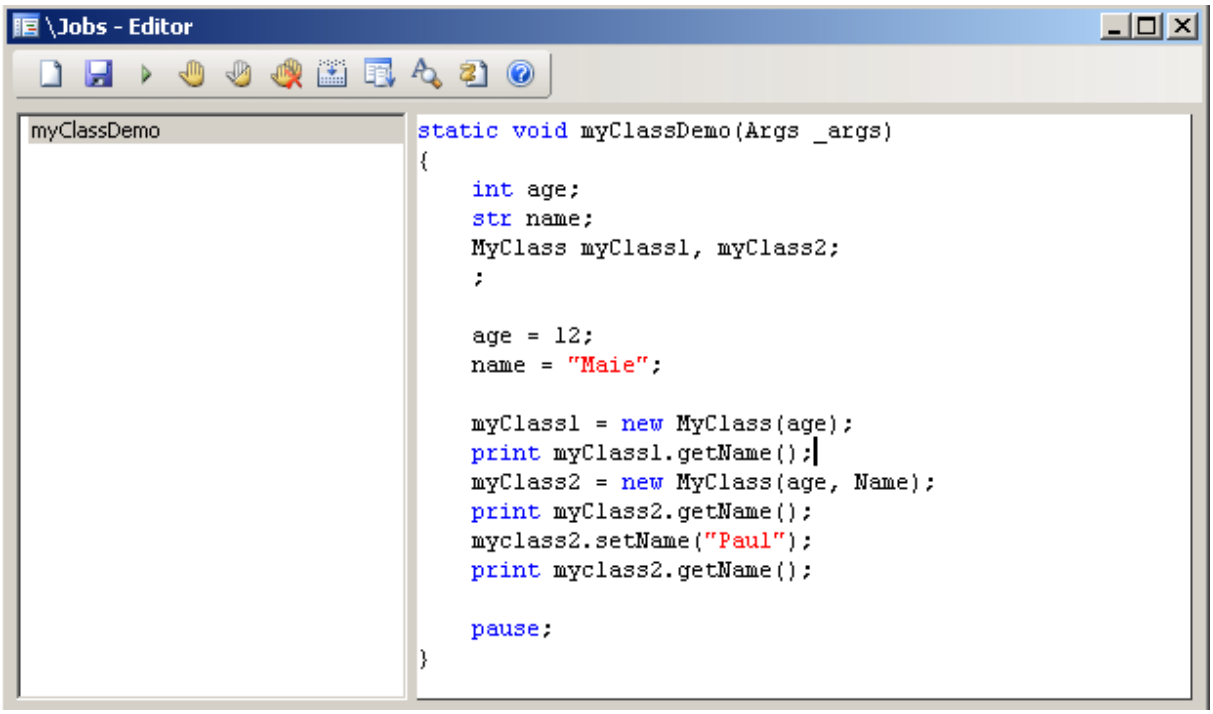
pause on meetod, mis jätab selle teateakna lahti, muidu sulguks see enne, kui me jõuaks teksti üldse näha.

Programmijupi käivitamiseks on rohelise noolega ikoon. Samuti saab käivitada klahviga F5. Avanema peaks nüüd järgnev vaade:



Joonis 20. Väljaprindi aken

Programm küsib nüüd, kas jätkata programmi tööd (jätkata koodi täitmist, mis on peale *pause;*) . Järgmiseks proovime demonstreerida varem loodud klassi MyClass. Selleks loome uue töö, nimega *myClassDemo*:



Joonis 21. Töö myClassDemo

Selles näites ja ka eelmistes (*MyClass* klassi meetodites) võis märgata *';'* märki üksinda real. See ei olnud kirjaviga. X++ keeles on meetodi sisu jagatud kahte ossa. Lokaalsete muutujate deklareerimine ning koodi täitmine. Nende kahe osa eraldamiseks ongi märk *';'*. Kui seda pole, siis kompilaator ei saa aru, kust lõppeb lokaalsete muutujate deklareermine. Käivitame koodi ning näeme järgmist pilti.

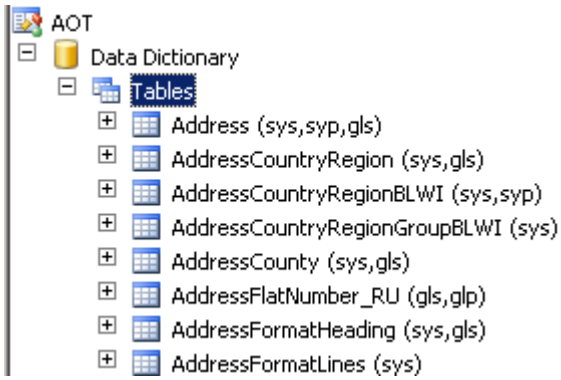
```
Malle
Maie
Paul
```

Joonis 22. Töö myClassDemo väljaprint

Esimene objekt *myClass1* loodi ainult ühe parameetriga *age*, mis tähendab, et klassi muutuja *name* omandab konstruktoris määratud algväärtuse "Malle" (vt. eelmist peatükki). Teine objekt *myClass2* loodi kahe parameetriga *age* ja *name*, mis tähendab, et parameetrina kaasa pandud *name* "Maie" saab objekti *name* väärtuseks. Kolmandaks katsetame meetodit setName(). Nagu väljatrükist näha, muutub objekti *name* väärtus "Paul"'iks.

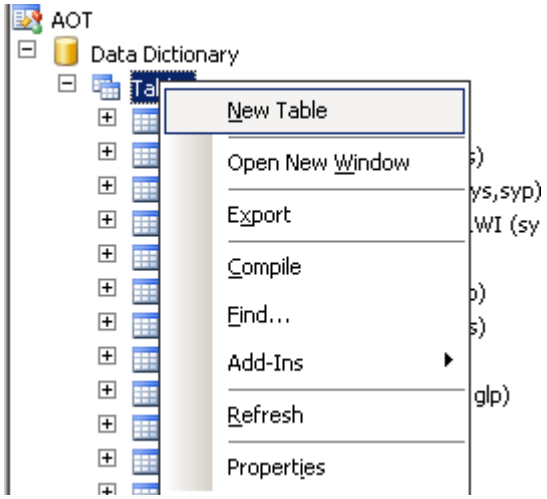
5. Tabelid

Tabelitest saab samuti tekitada objekte nagu klassidestki. Neid objekte nimetatakse tabeli puhvriteks (table buffer). Puhvrit deklareerides, luuakse instans, mis sisaldab endas kõiki kirjeid (records). Puhvri kaudu saab andmeid tabelist küsida, muuta, lisada ning kustutada. Tabelitele pääseb ligi:



Joonis 23. Tables asukoht AOT's

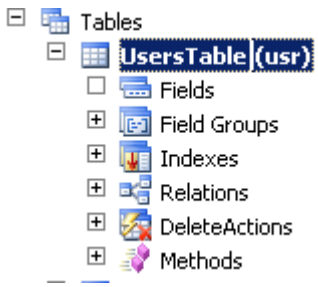
Loome uue tabeli, selleks klikime parema hiire klahviga jaotisel *Tables* ja valime *New Table*:



Joonis 24. Uue tabeli loomine

Paneme tabelile nimeks *UsersTable,* selleks klikime parema hiire klahviga loodud tabelil ja valime *Rename.*

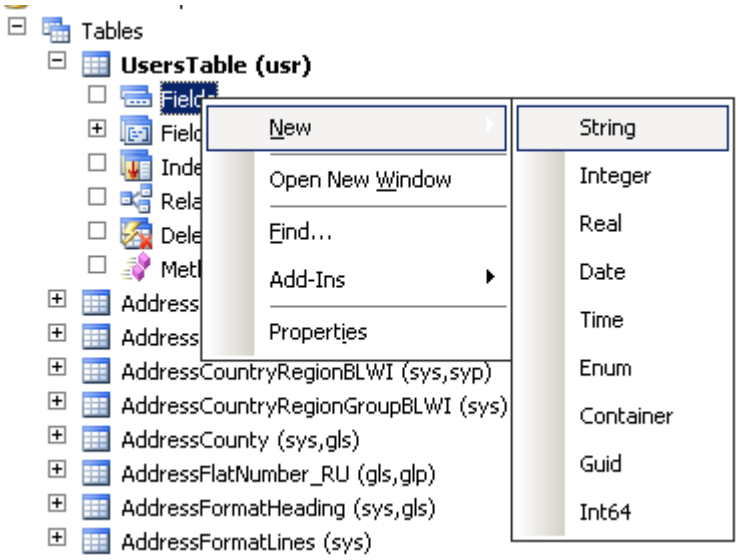
Vaatame, millest koosneb üks tabel:



Joonis 25. Tabeli komponendid

- *Fields* ehk väljad – siit saab lisada, muuta ja kustutada tabeli väljasid
- *Field Groups* ehk väljagrupid laseb väljasid grupeerida, et nende poole saaks pöörduda ja neid saaks kasutada grupina.
- *Indexes* – saab määrata indeksid.
- *Relations* – saab määrata seosed teiste tabelitega
- *DeleteActions* – saab määrata, mis juhtub, kui andmeid kustutatakse.
- *Methods* – saab lisada tabelile meetodeid.

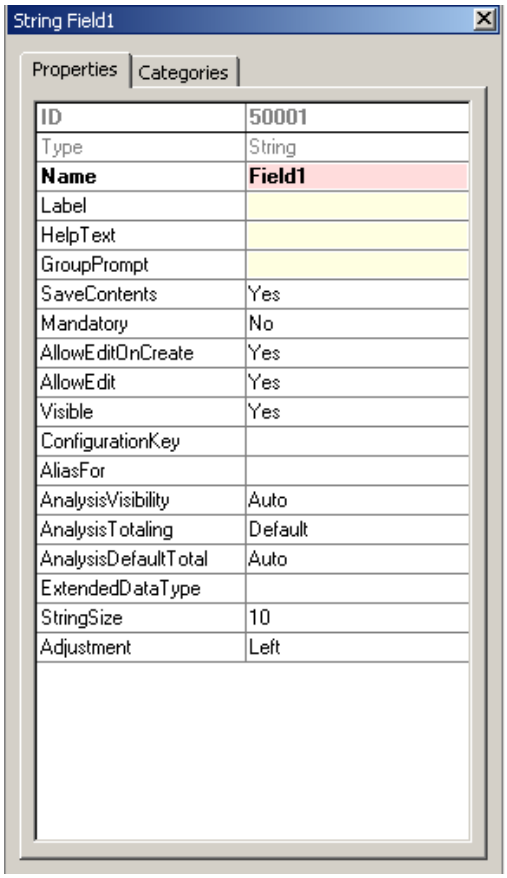
Lisame loodud tabelile mõned väljad. Selleks hiire parema klahviga klikime jaotilsele *Fields* ning valime *New -> String*:



Joonis 26. Uue String tüüpi välja lisamine tabelisse

Nagu näha, saab *New* jaotisest valida andmetüübi, mis tüüpi väli hakkab olema. Välja tüüpi hiljem muuta kahjuks ei saa, selleks tuleb vana väli kustutada ja uus lisada.

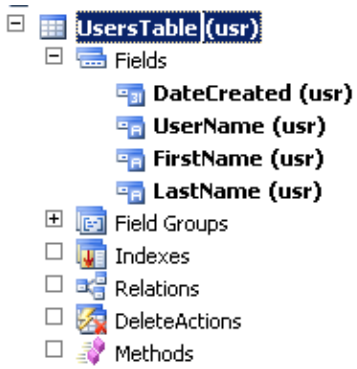
Nüüd peaks tekkima *Fields* jaotisesse väli nimega *Field1*. Vaatleme selle välja atribuute. Selleks valime parema hiire klahviga klikates *Properties*, millepeale avaneb aken:



Joonis 27. Tabeli välja atribuudid

Esimene asi, mis me atribuutides muudame, on *Name* (välja nimetus). Siia paneme *Field1* asemele *UserName*. Teine asi on *StringSize* (välja pikkus), milleks paneme 15.

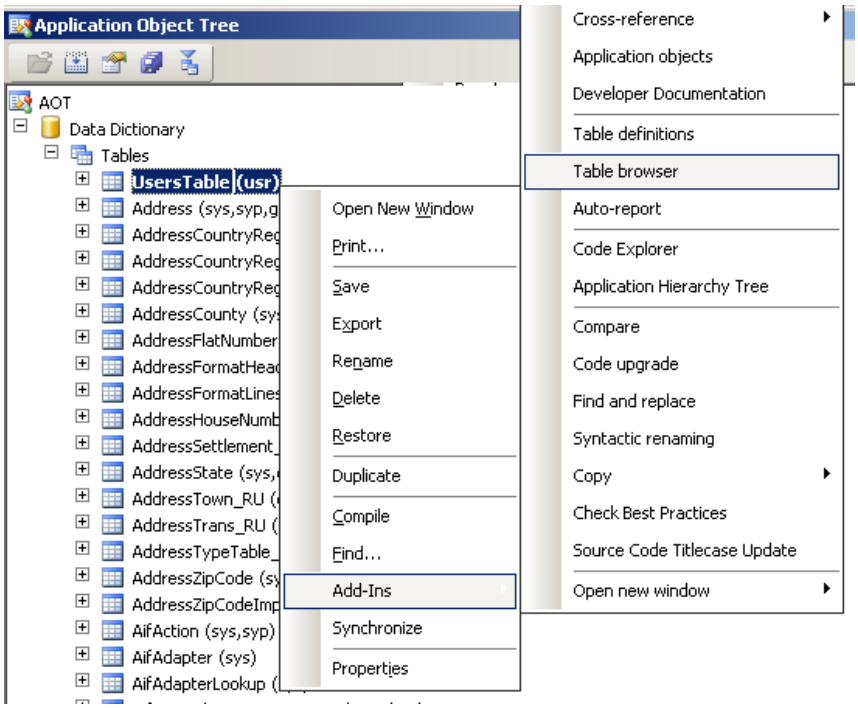
Analoogiliselt lisame kaks *string* tüüpi välja *FirstName* (eesnimi), *LastName* (perekonnanimi), pikkustega 20 ning ühe *date* tüüpi välja *DateCreated* (loomise kuupäev).



Joonis 28. UsersTable tabeli väljad

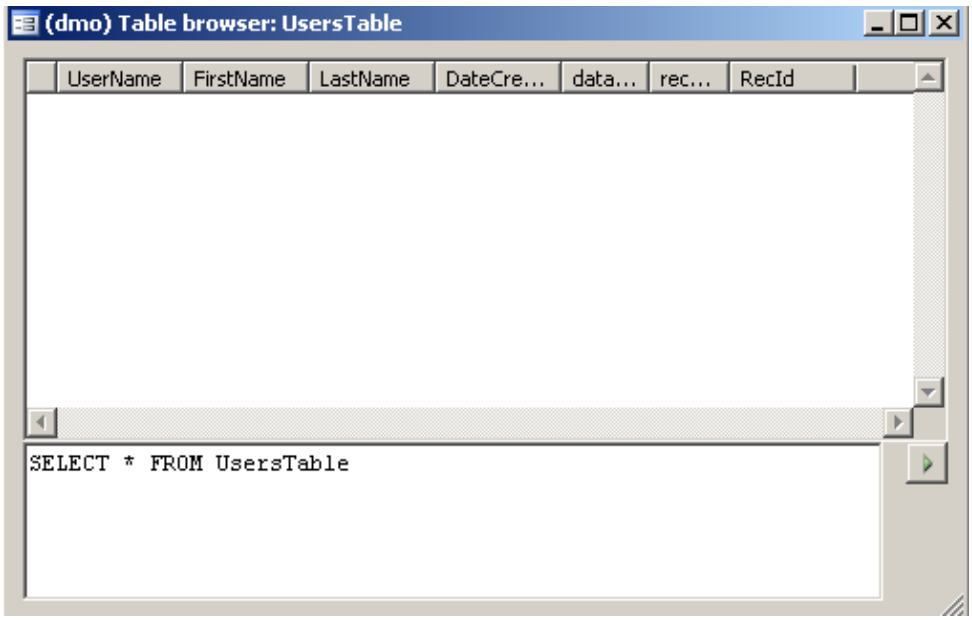
5.1 Tabelibrauser

Andmebaasi tabeli kirjete vaatamiseks/muutmiseks/lisamiseks/kustutamiseks on Microsoft Dynamics AX'is olemas tabelibrauser (*Table Browser*), mida saab käivitada klikates hiire parema klahviga soovitud tabeli peale ning valides *Add-Ins->Table browser:*



Joonis 29. Tabelibrauseri avamine

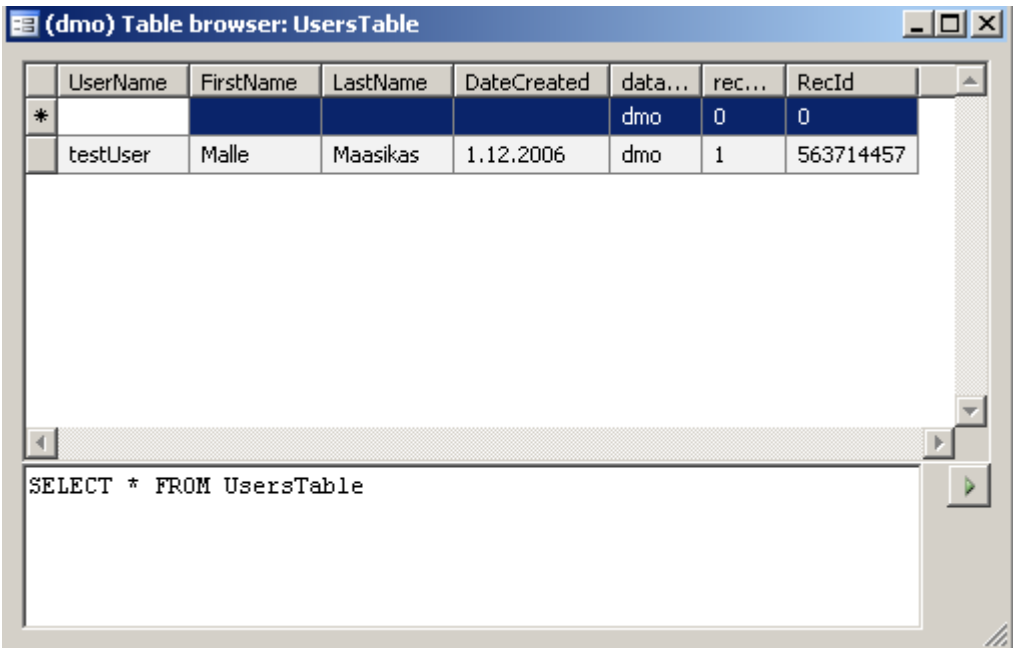
Avaneb järgmine aken:



Joonis 30. Tabelibrauser

See aken koosneb kahest osast. Ülemine osa on tabel, kus on kirjed ning alumine osa on koht, kuhu saab kirjutada X++ stiilis SQL päringuid.

Uue kirje lisamiseks tabelisse tuleb vajutada CTRL+N. Kirje salvestatakse CTRL+S ning kustutatakse ALT+F9



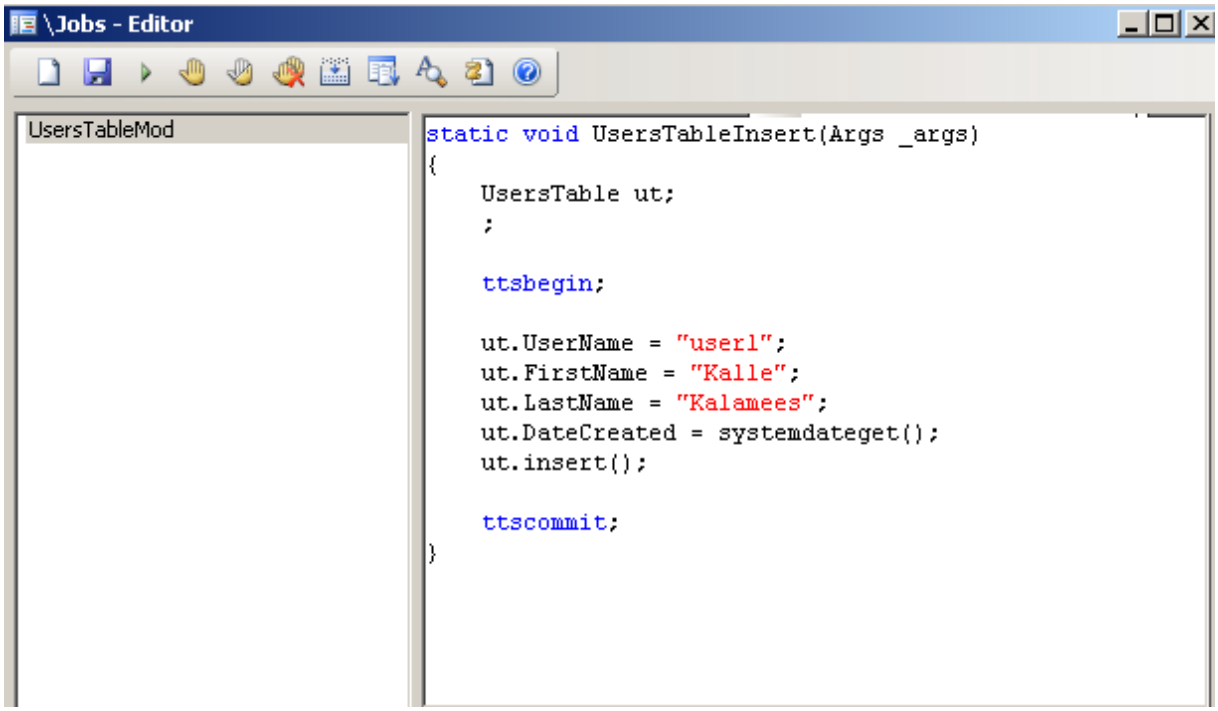
Joonis 31. Tabelibrauserisse kirje lisamine

5.2 Ligipääs tabelile läbi X++

Üks mitmest viisist tabeli poole pöörduda, on luua sellest puhver. Puhvri deklareerimine käib järgmiselt[4 lk 100]:

```
UsersTable usersTable;
```

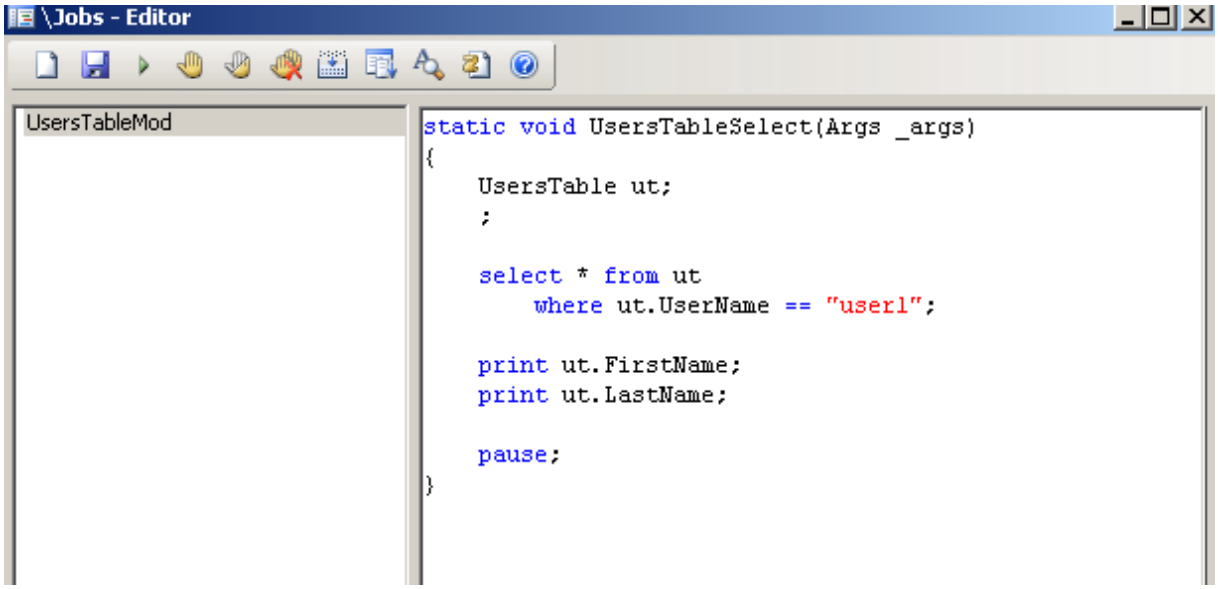
Teeme näiteks uue töö (job). Nimetame selle *UsersTableInsert.* Deklareerime *UsersTable* puhvri ja lisame sinna ridu. Ridade lisamise süntaks on järgmine:



Joonis 32. Tabelisse kirje lisamine läbi X++

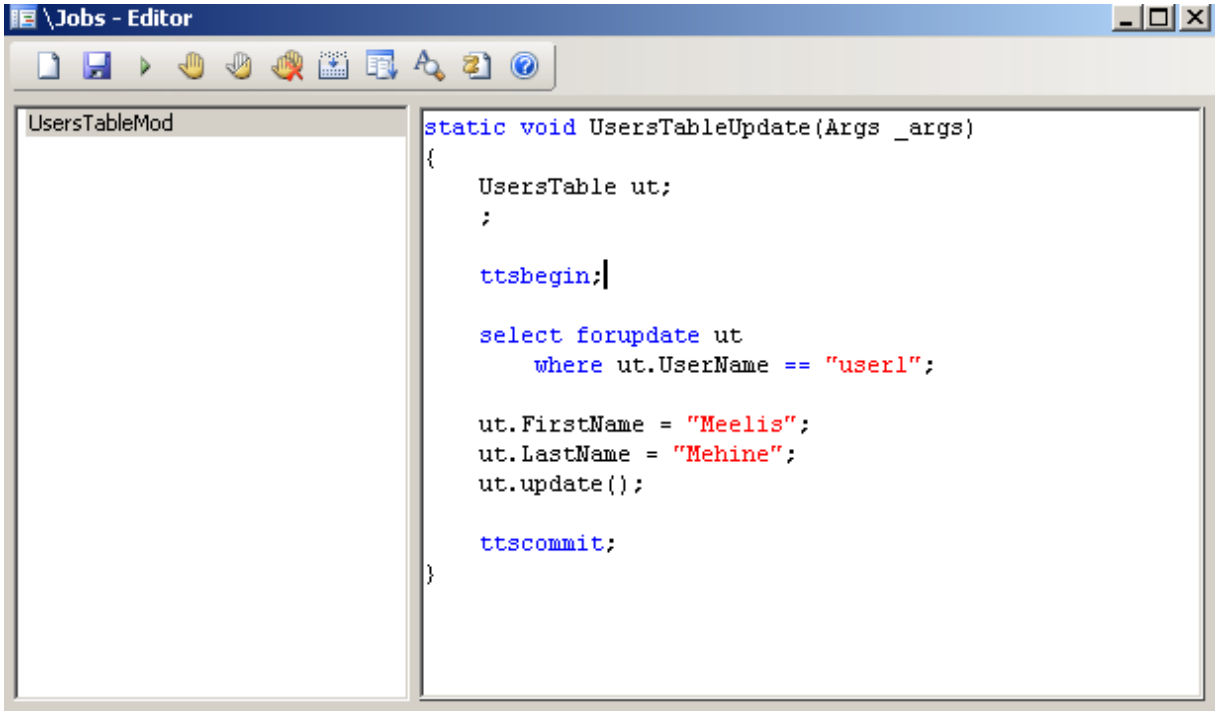
ttsbegin tähistab transaktsiooni algust ning *ttscommit* transaktsiooni lõppu. *systemdateget()* tagastab tänase kuupäeva. Kui töö käivitada, lisandub andmebaasi tabelisse *UsersTable* rida soovitud andmetega. Et selles veenduda, saab kasutada eelmises peatükis mainitud tabelibrauserit.

Kirjete küsimise süntaks on järgmine:

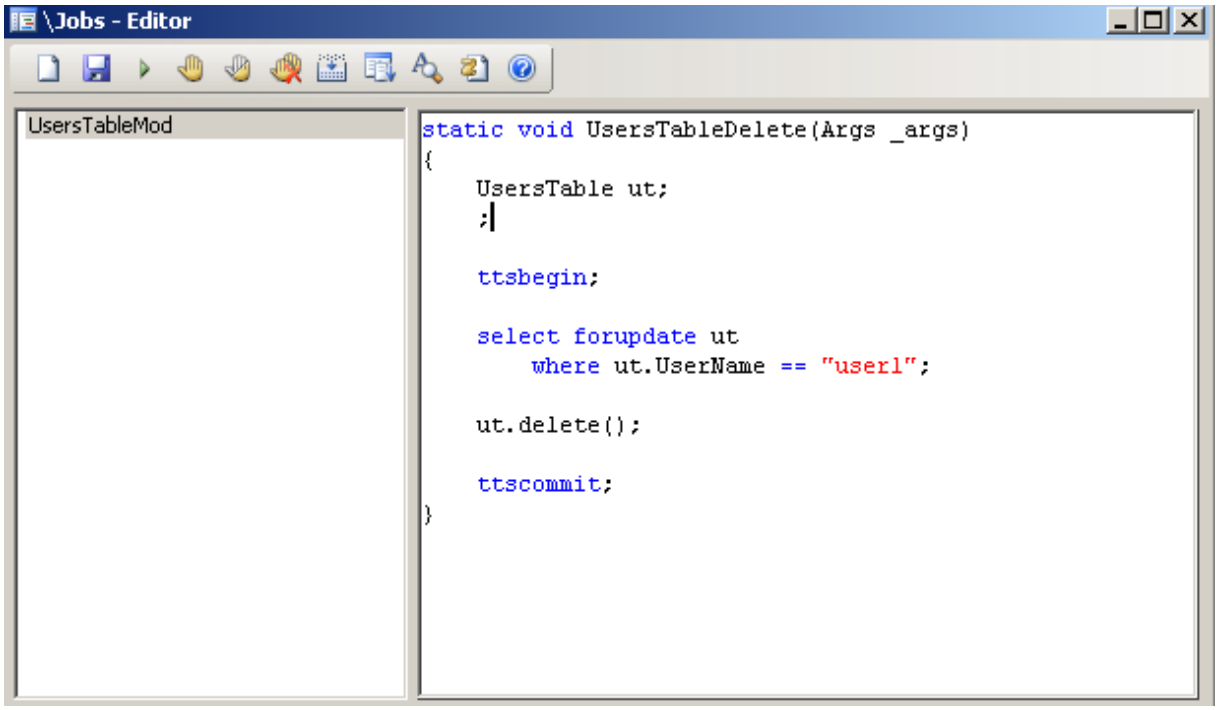


Joonis 33. Tabelist kirje küsimine läbi X++

Kirjete muutmise süntaks on järgmine:



Joonis 34. Tabelis kirje muutmine läbi X++

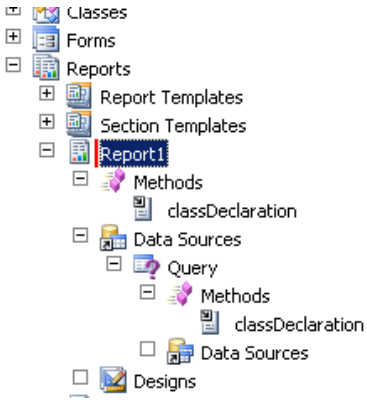


Joonis 33. Tabelist kirje kustutamine läbi X++

6. Aruanded

Aruandeid *(reports)* kasutatakse, et välja printida informatsiooni andmebaasi põhjal. [5 lk 168]

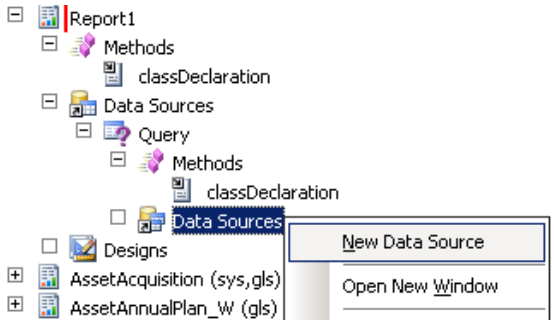
Aruannetele pääseb ligi AOT'st. Alustame sellest, et teeme uue aruande. Selleks klikime *Reports* jaotisele parema hiire klahviga ning valime *New Report*. Avaneb järgmine vaade:



Joonis 34. Aruande komponendid

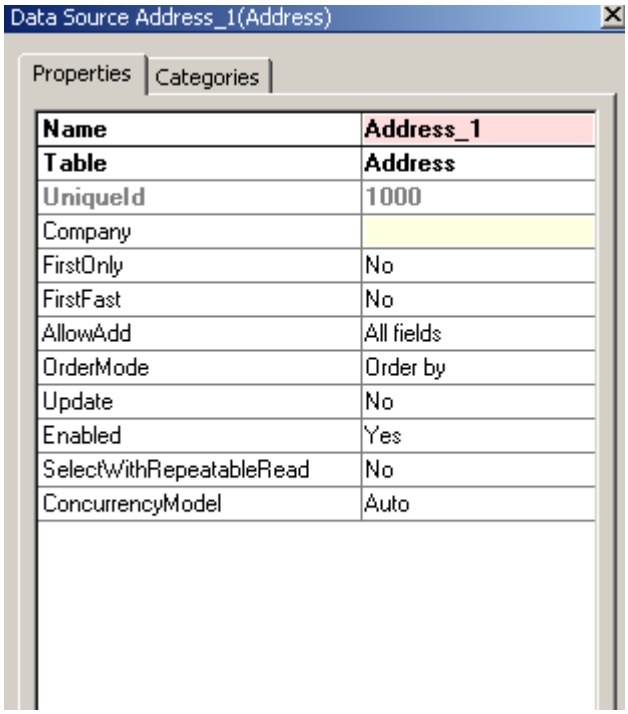
Siit on näha, et aruanne koosneb kolmest peamisest komponendist: meetodid *(Methods)*, andmeallikad *(Data Sources)* ning disain *(Design)*. Samamoodi nagu klassidel, on ka siin meetodites *classDeclaration*, kus kirjeldatakse ära globaalsed muutujad (kehtivad kõigis teistes alajaotistes).

Esimene asi, mis aruandel peab olema, on andmeallikas või andmeallikad (tabelid andmebaasis). Lisame ühe andmeallika *(Data Source)*. Selleks klikime parema hiire klahviga jaotisele *Data Sources* ning valime *New Datasource*.



Joonis 35. Uue andmeallika lisamine aruandele

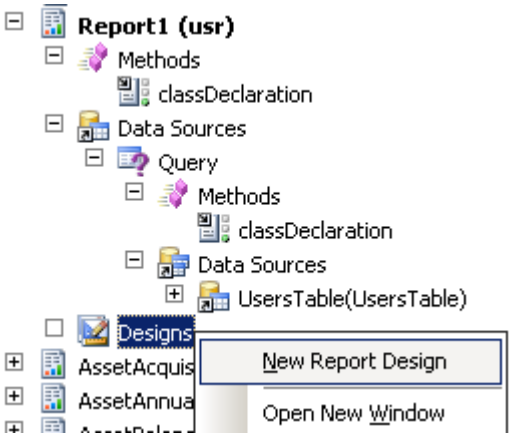
Atribuutide *(Properties)* aknas peaks avanema järgmine vaade (kui properties aken ei ole avatud, siis vali hiire parema klahviga properties):



Joonis 36. Andmeallika atribuudid

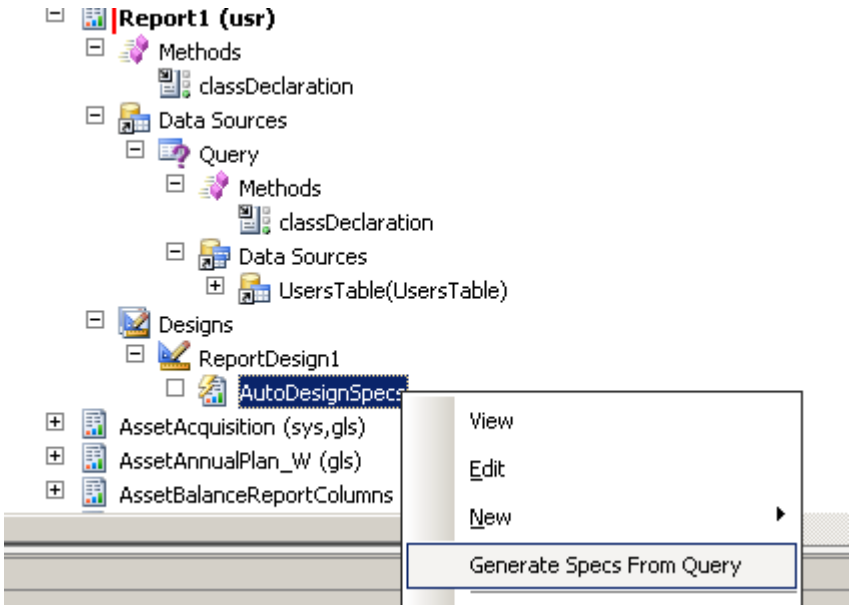
Siin on mitmeid andmeallika parameetreid, mida saab muuta. Meid huvitab hetkel väärtus *Table* (tabel mida kasutatakse andmeallikana). Siia paneme oma varem tehtud tabeli nime *UsersTable*. Peale igat muudatust CTRL+Shift+S, mis salvestab kõik muudatused.

Järgmiseks loome aruandele disaini. Selleks klikime hiire parema klahviga jaotisele *Design* ning valime *New Report Design.*



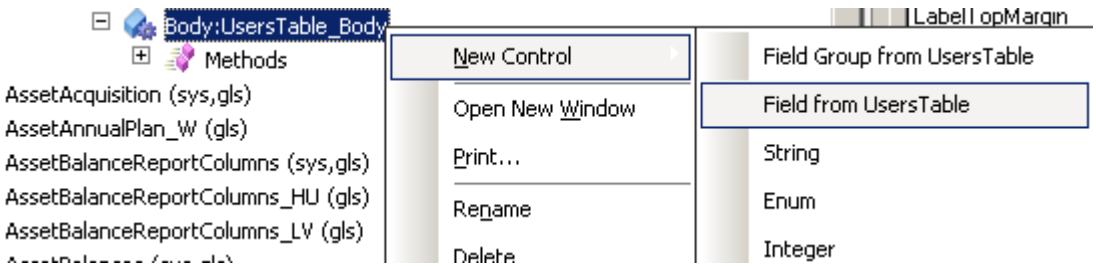
Joonis 37. Disaini loomine aruandele

Edasi klikime hiire parema klahviga *AutoDesinSpecs* peal ning valime *Generate Specs From Query*, mis loob disainibloki vastavalt *Data Sources* blokis olevatele tabelitele:



Joonis 38. Disainibloki genereerimine andmeallikatest

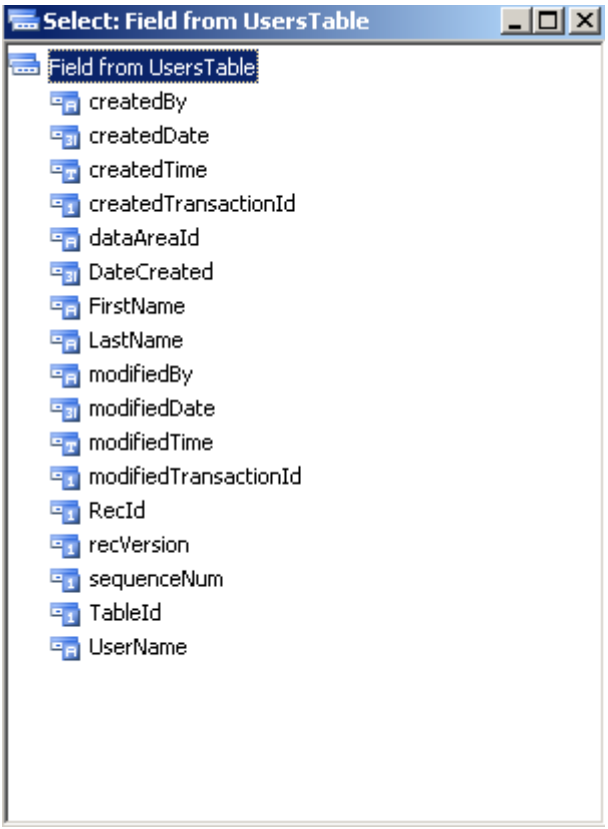
Nüüd lisame aruandele väljad, mida tahame kuvada. Selleks valime genereeritud *Body:UsersTable_Body*, klikime sellele hiire parema klahviga ning valime *New Control -> Field from UsersTable.* Mis tähendab, et tahame lisada välja tabelist UsersTable.



Joonis 39. Aruandele välja lisamine tabelist UsersTable

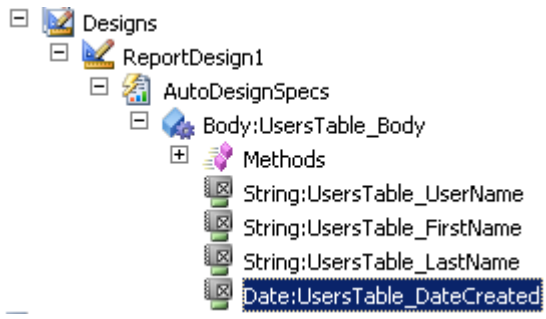
Sellest valikust üleval pool on valik *Field Group from UsersTable*. Kui tabelis on mingid väljad grupeeritud gruppi, saab terve grupi nüüd korraga lisada (vt. peatükk 6. Tabelid). Meie näites gruppi moodustatud pole, seega lisame väljad ükshaaval.

Nüüd peaks ilmuma nimekiri tabeli väljadest:



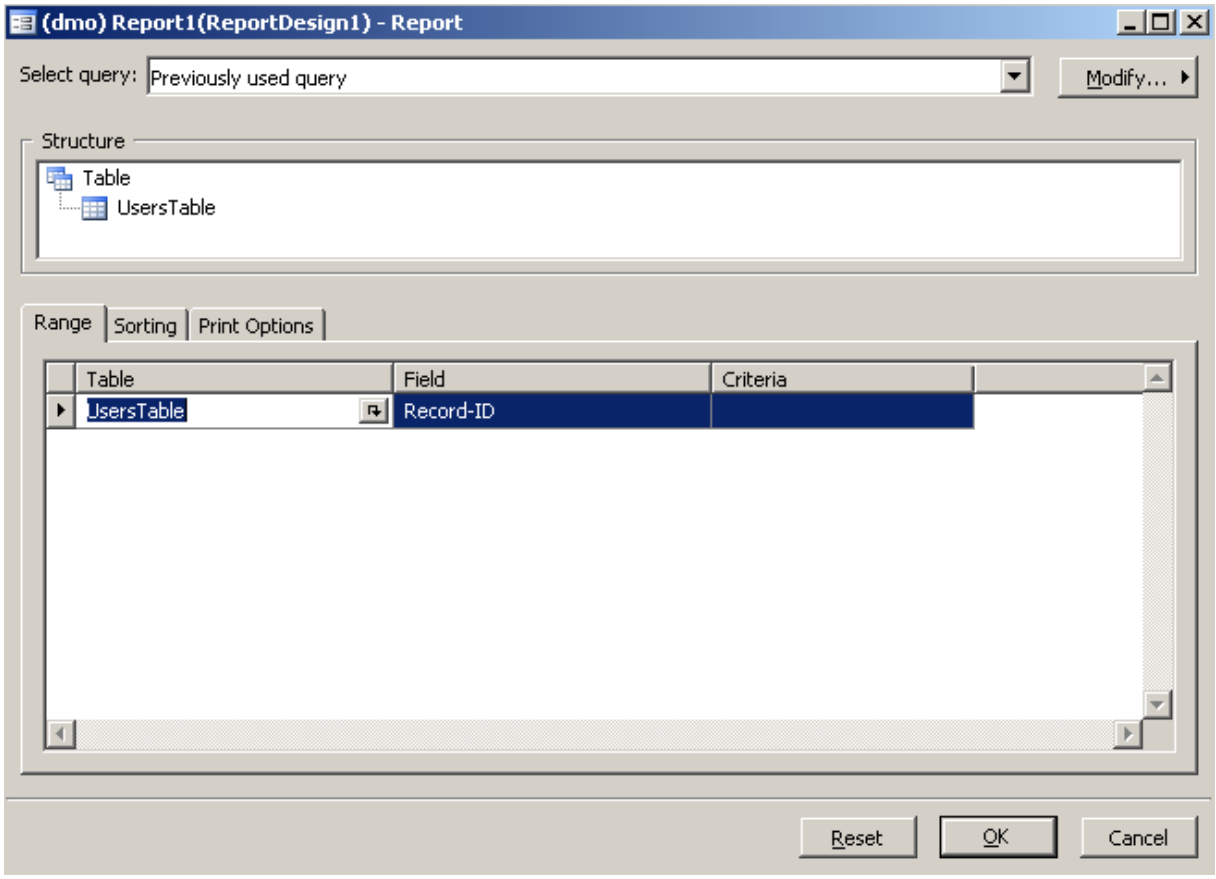
Joonis 40. Tabeli UsersTable väljad

Nagu nimekirjas näha, on siin palju selliseid välju, mida me ise tabelile lisanud pole. Need tekivad igale tabelile automaatselt. (väljad mis puudutavad kirje lisamise ja muutmise kuupäevi ning kellaaegu). Valime välja väljad mis meid huvitavad ning lohistame *(Drag-and-drop)* väljad ühekaupa jaotisesse *Body:UsersTable_Body*. Valime *UserName*, *FirstName*, *LastName*, *DateCreated*. *Disain* jaotis peaks nüüd välja nägema selline:



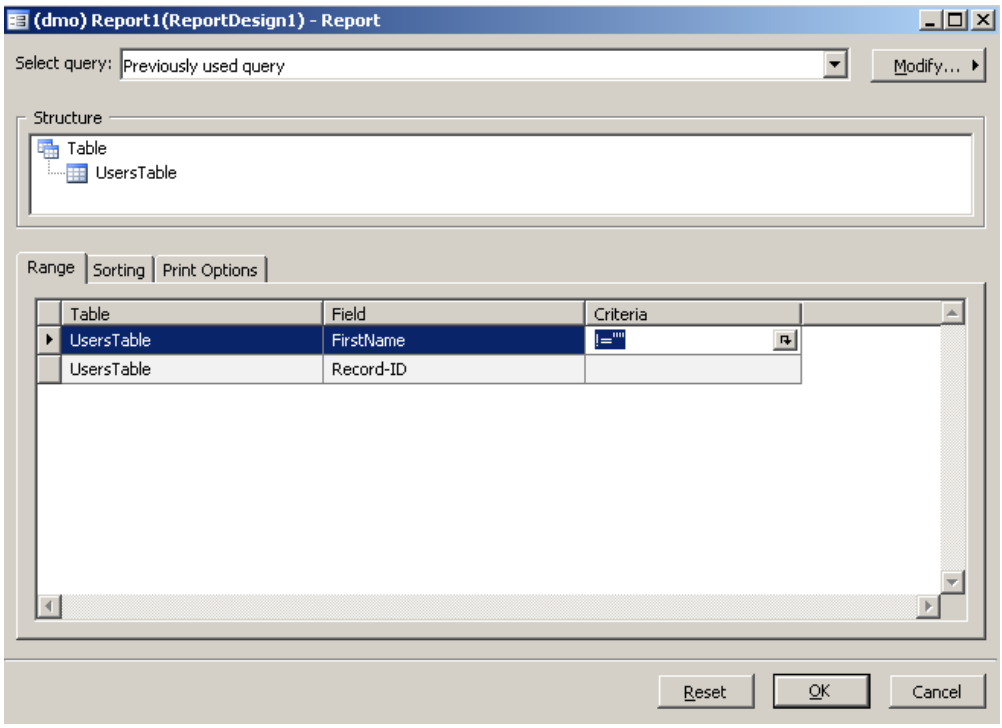
Joonis 41. Aruande disain jaotis pärast väljade lisamist

Lihtne aruanne on nüüd valmis. Käivitamiseks klikkame aruande peal *(Report1)* parema klahviga ning valime *Open*. Nüüd peaks avanema järgnev aken:



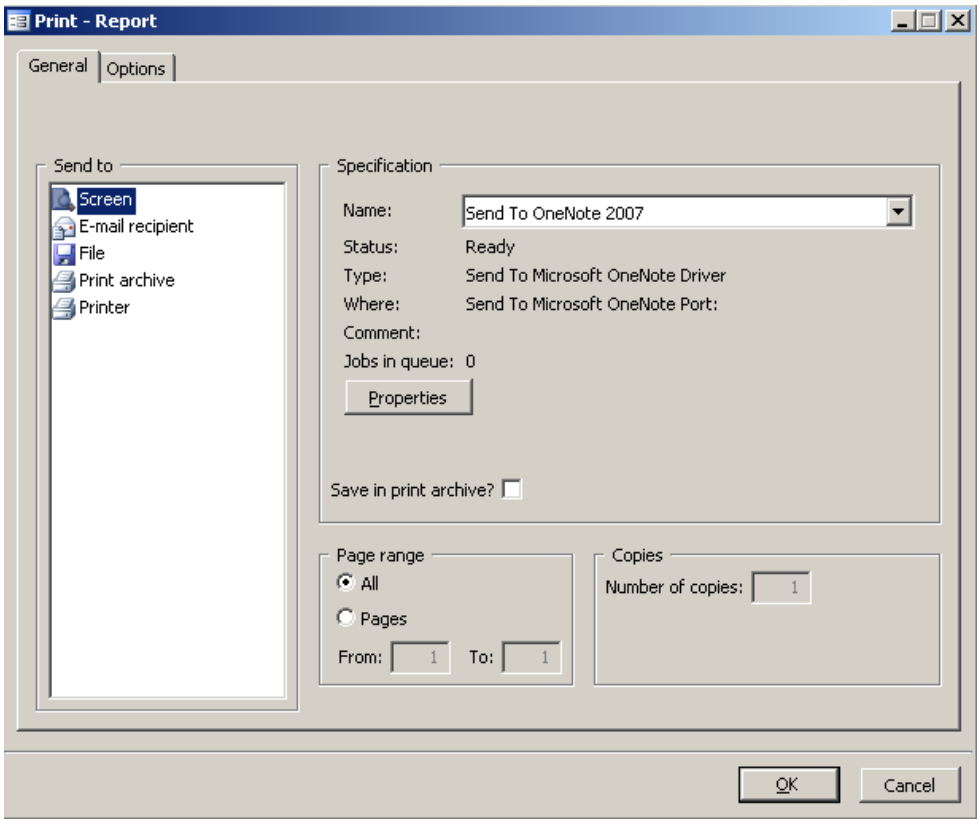
Joonis 42. Aken, kust saab valida filtreid aruandele

Avanev aken jaguneb kahte ossa: ülemises osas on andmeallikate struktuur (praegu on meil ainult üks tabel, kuid võimalik on ehitada väga keerulisi konstruktsioone erinevatest tabelitest) ja alumises osas saab määrata filtreid. Uue filtri lisamiseks tuleb vajutada CTRL+N. Lisame filtri, mis nõuab, et aruandel oleksid kõik kasutajad, kelle eesnimi ei ole tühi. Selleks valime *Field* lahtrisse *FirstName* ja *Criteria* lahtrisse kirjutame !="" (pole võrdne tühja stringiga). Valiku salvestame CTRL+S ning vajutame OK.



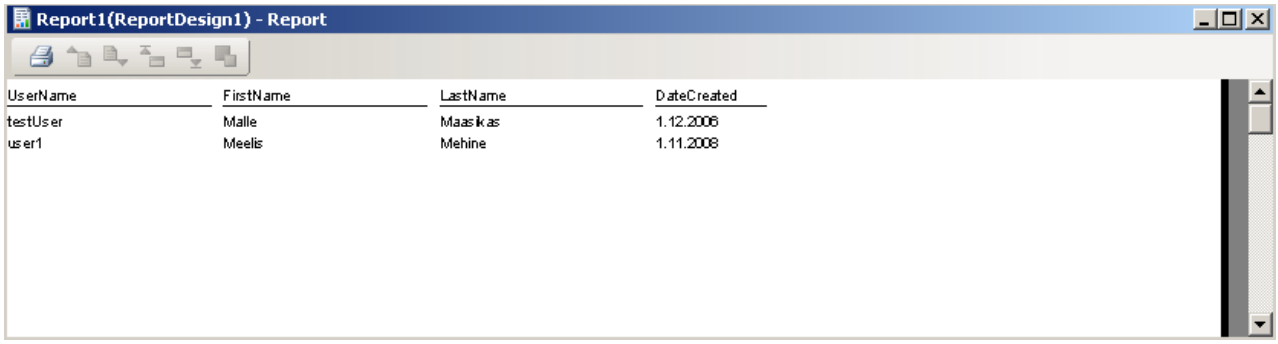
Joonis 43. Filtri lisamine aruandele

Järgmisena avaneb aken, kust saab valida, kuhu tuleb aruande väljatrükk:



Joonis 44. Aruande väljatrüki asukoha valimine

Microsoft Dynamics AX võimaldab aruannet salvestada faili (xml, pdf, xls), saata otse meilile (samuti pakub selleks erinevaid formaate), saata printerisse või ekraanile. Jätame vaikevaliku, milleks on *Screen* (ekraan) ning vajutame OK. Nüüd avaneb meile aruanne, mis näeb välja järgmine:



Kokkuvõte

Antud seminaritöös sai üle vaadatud Microsoft Dynamics AX mõned tähtsamad komponendid arendaja seisukohalt. Tutvust sai tehtud programmeerimiskeelega X++ ning sellele omapärase SQL päringute süntaksiga. Töös kasutatud näited olid tehtud võimalikud lihtsad, et demonstreerida põhifunktsionaalsust.

Puutumata jäi palju tähtsaid komponente, millest üks on Microsoft Dynamics AX põhikomponente – vormid (Forms). Vormide kaudu saab kasutaja andmeid sisestada, vaadata ning muuta.

Materjali Microsoft Dynamics AX kohta on väga vähe, veel vähem arendamise kohta. Huvi korral saab abi kasutatud kirjanduses nimetatud raamatutest[3, 4, 5]

Kasutatud kirjandus

1. A Brief History of DAX http://daxguy.blogspot.com/2006/12/brief-history-of-dax.html(2.11.2008)
2. ERP software: Microsoft Dynamics AX product information http://www.microsoft.com/dynamics/ax/product/default.mspx (2.11.2008)
3. Microsoft Dyamics™ AX 4.0 Cource 8623A: Development I Training: Microsoft Corporation, 2006
4. Microsoft Dyamics™ AX 4.0 Cource 8633A: Development II Training: Microsoft Corporation, 2006
5. Microsoft Dyamics™ AX 4.0 Cource 8645A: Development III Training: Microsoft Corporation, 2006
6. Kotta, P. Microsoft Dynamics AX logistikamoodul: Bakalaureusetöö. Tallinna Ülikool, Tallinn, 2008