L U K U 4

Käyttöliittymän luominen

Tässä luvussa

Tässä luvussa tutkitaan joitain tyypillisen Microsoft Windows -sovelluksen käyttöliittymän elementtejä. Opit, kuinka käsitellään sovelluksen valikkoja ja työkalurivejä ja kuinka lisätään koodi, joka suorittaa käyttäjän valinnan mukaiset tehtävät. Opit, kuinka saat käyttöliittymän toimimaan vuorovaikutuksessa käyttäjän kanssa. Opit, myös kuinka dialogeja käytetään sovelluksissa ja kuinka käytät dialogieditoria dialogimallien luomiseen.

Ennen kuin aloitat

Ennen kuin aloitat tämän luvun lukemisen sinun tulisi lukea luvut 2 ja 3, sekä suorittaa niissä olevat harjoitukset.

Oppitunti 1: Windows-sovelluksen käyttöliittymät

Tällä oppitunnilla tutkit käyttöliittymän elementtejä, jotka MFC AppWizard luo automaattisesti, kun teet sen avulla dokumentti/näkymä-pohjaisen sovelluksen. Opit, kuinka Microsoft Visual C++:n resurssieditoria käytetään käyttöliittymän standardiosien muuttamiseen omien vaatimustesi mukaisiksi, kuinka käsitellään valikoista ja työkaluriveiltä tehtyjä valintoja ja kuinka käyttöliittymää päivittämällä voit kommunikoida käyttäjän kanssa.

Tämän oppitunnin jälkeen:

- Tiedät, mitkä kuvakkeet on liitettävä sovellukseen, joka täyttää Microsoft Windows 98 tai Microsoft Windows NT -logovaatimukset ja kuinka suunnittelet kuvakkeita grafiikkaeditorilla.
- Tiedät, kuinka valikoita ja työkalurivejä muokataan.
- Tiedät, kuinka tehdään pikanäppäimiä valikoiden ja työkalurivien toiminnoille.
- Tiedät, kuinka sovellukseen lisätään koodi, joka huolehtii valikosta tai työkaluriviltä tehdyn valinnan mukaisen toiminnan suorittamisesta.
- Tiedät, kuinka muutat suorituksenaikaisesti valikon tai työkalurivin ulkoasua.
- Tiedät, kuinka tilariville kirjoitetaan.

Oppitunnin arvioitu kesto: 50 minuuttia

Sovelluksen ja dokumentin kuvakkeet

Windows 98 ja Windows NT -logovaatimusten mukaisten sovellusten tulee sisältää standardi- (32x32 pikseliä) ja pienikokoiset kuvakkeet (16x16 pikseliä) sekä sovellukselle että sen dokumenttityypille. Kuvakkeet ovat sovelluksen resursseja, jotka yhdistetään Windowsin kuvaketiedostoihin (.ico). MFC-sovellusrunko määrittelee automaattisesti oletuskuvakeresurssin ja luo kuvakkeet puolestasi.

Kaikilla sovelluksen resursseilla, kuten kuvakkeilla, valikoilla, dialogeilla, kontrolleilla ja niin edelleen, on ainutlaatuinen numeerinen määre, jota kutsutaan *resurssitunnisteeksi* (resource ID). Tiedostossa resource.h on määritelty helpommin muistettavat nimet, jotka vastaavat näitä tunnisteita.

Kuvassa 4.1 näet kuvakkeet, resurssi ID:t ja kuvaketiedostot, jotka on luotu MyApp-esimerkkisovellusta varten.

Icon	Resource ID	File
	IDR_MAINFRAME	MyApp\res\MyApp.ico
	IDR_MYAPPTYPE	MyApp\res\MyAppDoc.ico

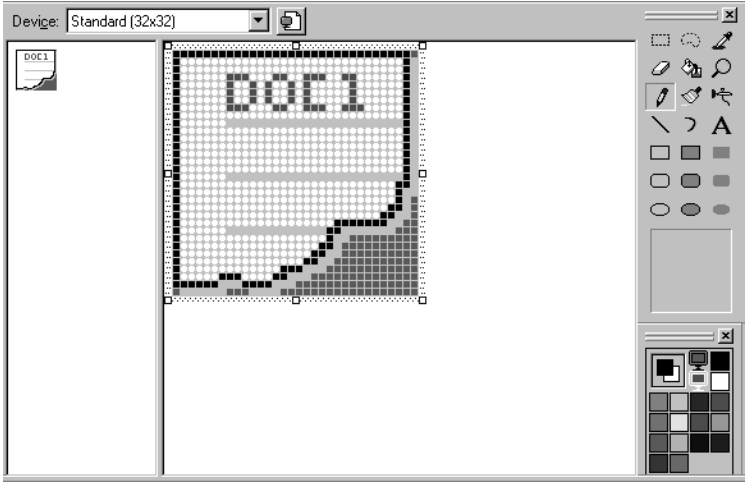
Kuva 4.1 MFC:n oletuskuvakkeet

Ohjelmasi sovellusolio rekisteröi kuvakkeet Windowsin tiedostonkäsittelijään kutsumalla **CWinApp::RegisterShellFileTypes()**-funktiota, kun sovellus käynnistyy.

Grafiikkaeditori

Visual C++:n mukana toimitettava grafiikkaeditori sisältää laajan kokoelman työkaluja kuvien piirtämistä varten. Sen lisäksi, että voit luoda ja muokata bittikarttatiedostoja (.bmp) voit käsitellä myös .gif ja .jpg -muodossa olevia kuvia ja muuttaa .gif ja .jpg -kuvia bittikartoiksi.

Grafiikkaeditoriin pääset avaamalla ResourceViewissä grafiikkaresurssin kaksoisnapauttamalla sitä. Näin pääset käsiksi erikoistyökaluihin, jotka tukevat kuvakkeita ja työkalurivin bittikarttoja. Kuvassa 4.2 näet grafiikkaeditorin, johon on avattu standardikokoinen vakiodokumenttikuvake.



Kuva 4.2 Grafiikkaeditori

► **Vakiokuvakkeiden muokkaaminen**

1. Palaa MyApp-projektiin, jota muokkasit luvussa 3.
2. Avaa MyApp **Resources**-oksa **ResourceView**issä.
3. Avaa Icon-kansio.
4. Avaa kuvake grafiikkaeditoriin kaksoisnapauttamalla **IDR_MAINFRAME** -objektia.

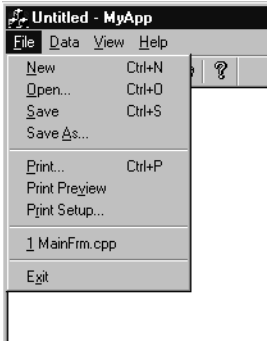
Kokeile grafiikkaeditorin piirtotoimintoja. Huomaa, että voit vaihtaa normaalin Standard (32x32) kuvakekoon ja pienen Small (16x16) kuvakekoon välillä käyttämällä vasemmassa yläkulmassa olevaa **Device**-valikkoa.

Voit käyttää **New Device Image** -toimintoa muunmuotoisten kuvakkeiden luomiseen. Grafiikkaeditorissa on käytettävissä erilaisia kynä- ja pensselityylejä, alueen täyttöjä ja tekstityökaluja; voit myös valita osan kuvakkeesta ja siirtää sen toiseen paikkaan, muuttaa sen kokoa tai kopioida sen leikepöydälle.

Jos tunnet itsesi luovaksi, voit suunnitella oman kuvakkeen MyApp-sovellukselle. Helpoin tapa oman kuvakkeen tekemiseksi on valmiin kuvakkeen muokkaaminen, jolloin resurssitunnistetta ei tarvitse muuttaa. Kun käännät sovelluksen, .ico tiedostoon tekemäsi muutokset tallentuvat.

Sovelluksen valikkojen muokkaaminen

Valikot ovat ohjelmoijalle mukava ja yhdenmukainen tapa ryhmitellä komentoja ja käyttäjille helppo tapa päästä niihin käsiksi. MFC AppWizardilla luotu sovellus sisältää yleensä ohjelmoijan tekemien valintojen mukaisen päävalikon. Kuvassa 4.3 on tyypillinen SDI-sovellukseen luotu valikko ja joukko valikon käyttöliit-tymän elementtejä.

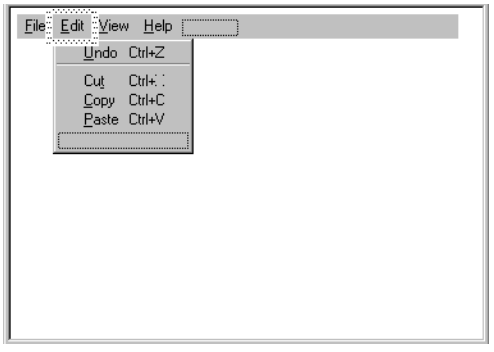


Kuva 4.3 SDI-sovelluksen valikko

Valikkoon kokonaisuutena viitataan resurssitunnisteella **IDR_MAINFRAME**. Huomaat, että tunniste on sama kuin sovelluksen kuvakkeella, josta luit luvussa

3. Kun sovelluksen malliobjekti on luotu, se liitetään resurssitunnisteeseen, joka määrittelee joukon kyseiseen dokumenttityyppiin liittyviä resursseja. Näihin voivat kuulua valikko, työkalurivi, kuvake, pikanäppäintaulukko ja merkkijono-resurssit.

Visual C++:n mukana toimitettavaa menueditoria (kuvassa 4.4) käyttämällä voit rakentaa valikon visuaalisesti ja muokata valikkokomentojen ominaisuuksia.



Kuva 4.4 Menueditori

Seuraavassa harjoituksessa opit, kuinka valikoita lisätään ja poistetaan ja lisäät komentoja MyApp-sovelluksen valikoihin. Aloitat poistamalla **Edit**-valikon, jonka näet kuvassa 4.4. Tämä on tarpeen, jos sovelluksesi ei tue objektien valitsemista tai **Cut**, **Copy** ja **Paste** -komentoja, jotka toimivat valituille objekteille. Tälle paikalle lisäät uuden **Data**-valikon. Data-valikkoon tulevat komennot, joiden avulla käyttäjä ottaa yhteyden tietokantaan ja hakee sovelluksen tiedot. Valikkoon lisätään kaksi komentoa; **Connect** ja **Upload**.

► **Edit-valikon poistaminen**

1. Napauta **ResourceView**-välilehteä MyApp-projektissa ja avaa MyApp Resources-kansio.
2. Avaa Menu-kansio.
3. Avaa menueditori kaksoisnapauttamalla **IDR_MAINFRAME**-valikkoresurssia.
4. Napauta kohtaa **Edit** valikkopalkissa ja paina DELETE. Hyväksy Edit-valikon poistaminen naputtamalla **OK**.

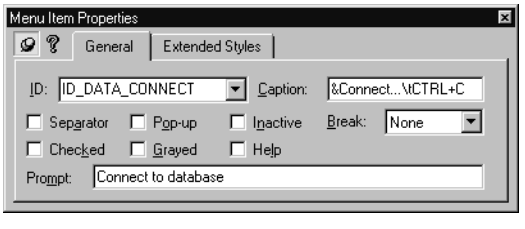
► **Data-valikon lisääminen**

1. Napauta tyhjää kohtaa heti **Help**-valikon oikealla puolella.
2. Raahaa tyhjä komento **File** ja **View** -valikkojen väliin.
3. Ota tyhjän komennon ominaisuudet näkyviin kaksoisnapauttamalla sitä. Kirjoita **Caption**-ominaisuudeksi **&Data**. &-merkki määrää, että sitä seuraava kirjain toimii valikon valintanäppäimenä. Valintanäppäintä käytetään valikon valitsemiseen näppäimistöltä. **Data**-valikon valintanäppäimeksi on merkitty "D", joten käyttäjä voi valita **Data**-valikon painamalla ALT+D.

Huomio Jos haluat sijoittaa &-merkin valikon nimeen käytä kahta &-merk-kiä peräkkäin. Esimerkiksi "Mutt && Jeff" näkyy valikon nimenä muodossa "Mutt & Jeff".

► **Connect-komennon lisääminen Data-valikkoon**

1. Jos **Data**-valikko ei ole valittuna, valitse se napauttamalla hiirellä.
2. Napauta **Data**-valikon otsikon alla olevaa tyhjää komentoa.
3. Kirjoita merkkijono **&Connect…\tCTRL+C**.
   - **\t** on sarkainmerkki.
   - **CTRL+C** viittaa seuraavassa harjoituksessa valikkokomentoon liitettävään pikanäppäimeen.
4. Kun aloitat kirjoittamisen, ilmestyy **Menu Item Properties** -dialogi näkyviin. Kirjoita **Prompt**-ruutuun **Connect to database**. Tämä teksti tulee näkyviin tilariville, kun komento on valittuna.
5. Napauta **Menu Item Properties** -dialogin vasemmassa yläreunassa olevaa nastakuvaketta. Näin dialogi pysyy auki liikkuessasi valikkokomennosta toiseen.
6. Napauta toista komentoa ja sitten uudelleen komentoa **Connect**. Dialogi-ikkunan tulisi nyt näyttää samanlaiselta kuin kuvassa 4.5.



Kuva 4.5 Menu Item Properties -dialogi

7. Huomaa, että editori on muodostanut komennontunnisteen **ID_DATA_CONNECT** lisäämällä komennon nimen valikon nimeen. Voit muuttaa tunnisteen, mutta yleensä säilytetään oletustunniste.

Jokainen valittavissa oleva valikkokomento (kaikki muut kuin valikko-otsikot ja erotinviivat) on liitetty tunnisteeseen, jonka avulla se voidaan yhdistää toiminnon suorittavaan funktioon. Muista, että kun valikkokomento on valittu, sovellusrunko lähettää WM_COMMAND-sanoman, jonka käsittelee yksi sovelluksen objekteista. Tämä sanoma ottaa komennon tunnisteen parametriksi. Opit, kuinka komentotunniste liitetään käsittelijäfunktioon myöhemmin tässä luvussa jaksossa *Valikko- ja työkalurivivalintojen käsitteleminen*.

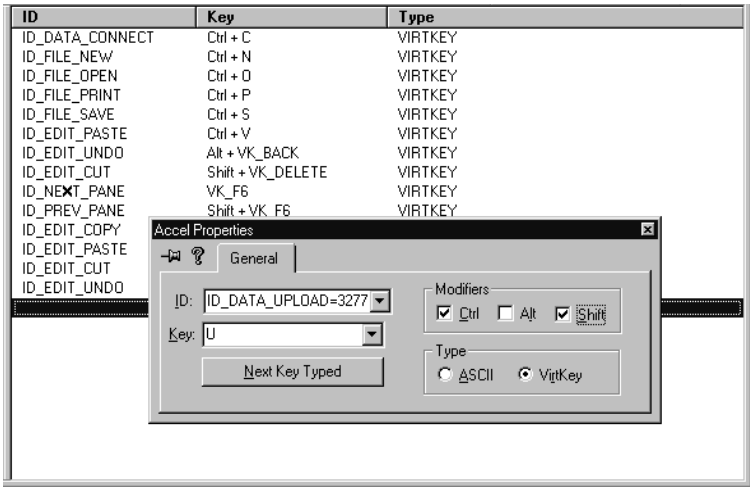
► **Upload-komennon lisääminen Data-valikkoon**

Lisää uusi komento **Data**-valikkoon edellä kuvatulla tavalla. Tämän komennon otsikoksi (caption) kirjoitetaan **&Upload...\tCTRL+U** ja kohtaan Promt tulee teksti **Upload data to the database**. Anna menueditorin muodostaa komentotunniste **ID_DATA_UPLOAD**.

Pikanäppäinten lisääminen

Kuten muistat, **Data**-valikon komentojen otsikoissa viitattiin näppäimiin, joita voidaan käyttää pikanäppäiminä — siis **Connect**-komennolla CTRL+C ja CTRL+U **Upload**-komennolla. Pikanäppäimiä käytetään näppäimistöoikoteinä sovelluksen komentoihin, jotka ovat valittavissa myös valikoista tai työkaluriveiltä. Voit tosin määritellä pikanäppäimen myös toiminnolle, johon ei liity yhtään käyttöliittymän objektia.

Pikanäppäimet, englanniksi shortcut keys tai alunperin accelerator keys, määritellään *pikanäppäintaulukossa* (accelerator table), joka on listan pikanäppäimistä ja niihin liittyvistä komennoista sisältävä Windows-resurssi. Taulukkoa voidaan muokata käyttämällä accelerator-editoria (kuvassa 4.6).



Kuva 4.6 Accelerator-editori

Tässä harjoituksessa lisäät CTRL+C ja CTRL+U -näppäinyhdistelmät sovelluksen pikanäppäintaulukkoon editoria käyttämällä.

► **Pikanäppäimen määrittäminen**

1. Avaa Accelerator-kansio Resource Viewissä.
2. Avaa editori kaksoisnapautamalla **IDR_MAINFRAME** pikanäppäinresurssia.

   Huomaa, että CTRL+C näppäinyhdistelmä on jo yhdistetty tunnisteeseen **ID_EDIT_ COPY**. Pääset muuttamaan kohdetta kaksoisnapauttamalla **CTRL+C** merkintää taulukon yläosassa.
3. Valitse **ID_DATA_CONNECT Accel Properties** -dialogin **ID**-listasta tunnisteen päivittämistä varten.
4. Lisätäksesi uuden pikanäppäimen, kaksoisnapauta listan lopussa olevaa tyhjää paikkaa. Napauta **ID_DATA_UPLOAD** kohtaa **ID**-listasta.
5. Kirjoita **u Key**-ruutuun.
6. Valitse **Ctrl**-valintaruutu ja poista valinta Shift-valintaruudusta. Editorin ikkunan tulisi näyttää samanlaiselta kuin kuvassa 4.6.
7. Sulje **Accel Properties** -dialogi. Pikanäppäin on lisätty taulukkoon.

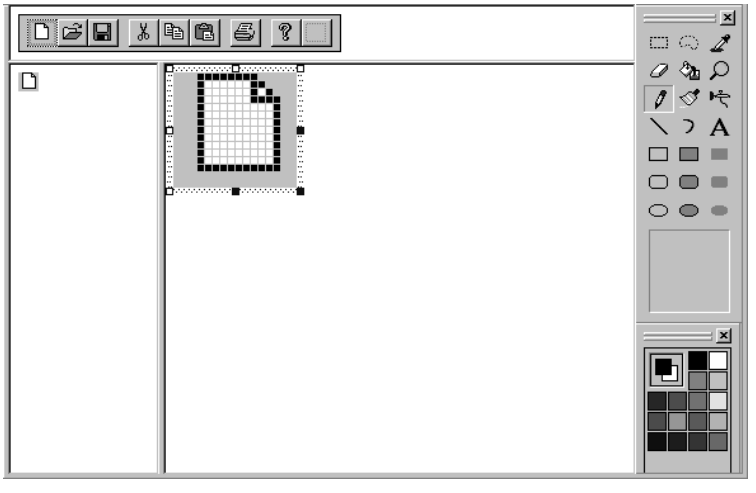
Sovelluksen työkalupalkkien muokkaaminen

Kun AppWizard luo sovelluksen, se lisää siihen standardin työkalurivin, jossa on **File** ja **Edit** -valikkojen käytetyimpiin komentoihin liitetyt painikkeet. Työkalupalkki kuuluu resurssiryhmään, joka on yhdistetty sovelluksen dokumenttimalliin, joten sillä on sama **IDR_MAINFRAME-**tunniste kuin sovelluksen kuvakkeella, pikanäppäimillä ja valikkoresursseilla.

Työkalupalkkiresurssi on liitetty bittikarttatiedostoon, joka sisältää painikkeiden kuvat. **IDR_MAINFRAME**-työkalupalkkiin liitetty tiedosto on nimeltään toolbar.bmp. Sovellusrunko sijoittaa kopion tästä tiedostosta \res-kansioon resurs-sikansion alle. Jos luot lisää työkalupalkkeja, lisäbittikartat tallennetaan resurssi-tunnisteita muistuttavilla nimillä tähän samaan kansioon.

Kaikkien painikkeiden kuvien tulee olla saman kokoisia. Oletuskoko on leveyssuunnassa 16 pikseliä ja pystysuunnassa 15 pikseliä. Bittikarttatiedosto sisältää kaikki kuvat siinä järjestyksessä kuin ne esiintyvät vastaavassa resurssimäärittelyssä. Tämän takia on suositeltavaa, että muokkaat tätä tiedostoa käyttämällä grafiikkaeditorin erikoisominaisuuksia. Editoriin pääset kaksoisnapauttamalla työkalupalkkiobjektia Resource View -ikkunassa. Nämä ominaisuudet antavat sinun muokata bittikarttaa hallittavissa olevissa, painikkeen kokoisissa paloissa ja ylläpitää painikkeiden ja komentotunnisteiden väliset yhteydet.

Kuvassa 4.7 näet grafiikkaeditorin, jota käytetään työkalupalkin muokkaamiseen.



Kuva 4.7 Työkalurivin muokkaaminen grafiikkaeditorilla

Tässä harjoituksessa poistat **Cut**, **Copy** ja **Paste** -painikkeet työkalupalkista.

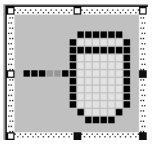
► **Työkalupalkin painikkeen poistaminen**

1. Avaa MyAppin resurssit ResourceViewissä.
2. Avaa Toolbar-kansio ja avaa työkalupalkkieditori kaksoisnapauttamalla **IDR_MAINFRAME** työkalupalkkiresurssia.
3. Raahaa **Cut**-painike pois työkaluriviltä.
4. Tee samoin **Copy** ja **Paste** -painikkeille.

Tässä harjoituksessa lisäät aiemmin lisättyjä **Connect**ja **Upload** -komentoja vastaavat painikkeet työkalupalkkiin.

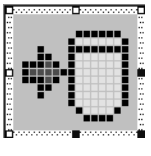
► **Painikkeen lisääminen työkaluriville**

1. Napauta työkalupalkin oikeassa reunassa olevaa tyhjää painiketta.
2. Raahaa painike haluttuun paikkaan työkalupalkissa. Tässä harjoituksessa siirretään painike **Print**-painikkeen vasemmalle puolelle.
3. Käytä grafiikkaeditorin toimintoja yksinkertaisen kuvakkeen suunnittelemiseen. Kuvassa 4.8 on malli, joka esittää tietokantaan yhdistämistä.



Kuva 4.8 Connect Toolbar -painike

4. Paina ENTER. **Toolbar Button Properties** -dialogi avautuu.
5. Napauta kohtaa **ID_DATA_CONNECT ID**-luettelossa. Liitä uusi tunniste napauttamalla toista työkalupalkin painiketta.
6. Palauta Connect-painikkeen ominaisuudet näkyviin napauttamalla työkalurivin Connect-painiketta. Huomaa, että **ID_DATA_CONNECT-**komentotunnisteeseen liitetty seliteteksti tulee näkyviin. Määritä painikkeen vihjeteksti lisäämällä selitteen loppuun \nConnect-teksti.
7. Toista edelliset toimenpiteet toiselle työkalupalkin painikkeelle lisätäksesi kuvassa 4.9 näkyvän painikkeen **Upload**-komentoa varten. Liitä tämä painike komentotunnisteeseen ID_DATA_UPLOAD ja määritä vihjeteksti lisäämällä teksti \nUpload.



Kuva 4.9 Upload-painike

8. Käännä ja käynnistä MyApp-sovellus. Kokeile muutoksia, jotka teit valikkoon ja työkalupalkkiin. Huomaa, että lisäämäsi komennot näkyvät harmaina. Tämä johtuu siitä, että komennoille ei ole vielä määritelty käsittelijöitä. Voit kuitenkin kokeilla, että seliteteksti ja vihjeteksti näkyvät oikein.

Valikko- ja työkalupalkkivalintojen käsittely

Luvun 3 oppitunnilla 3 opit, kuinka ClassWizardia käytetään käyttäjän valikoista tai työkalupalkista tekemien valintojen aiheuttamien sanomien käsittelyyn. Seuraavassa harjoituksessa lisäät käsittelijät **Connect** ja **Upload** -komennoille.

MFC-sovellusrungon komentojen reititysominaisuus mahdollistaa käsittelijän sisällyttämisen luokkaan, jonka tehtävää lähinnä käsittelijän tekemä toiminto on. Voit päätellä, että yhteys etätietokantaan mieltyy sovellukseen kokonaisuudessa, jolloin **Connect**-komennnon käsittelijä on parasta sijoittaa **CMyAppApp**-sovellusluokkaan.

Tässä tapauksessa **Connect**-komennon käsittelijä sisältää koodin, jonka avulla paikannetaan ja muodostetaan yhteys etätietokantaan. Kannattaa varmaankin tallettaa tietokantayhteyden nykyinen tila boolean-muuttujaan niin, että sovellus pystyy aina kertomaan, onko yhteys olemassa vai ei. Tämän muuttujan arvoksi asetetaan TRUE onnistuneen yhteydenoton jälkeen ja FALSE yhteyden purkamisen tai epäonnistuneen yhteydenoton jälkeen. Esimerkkiämme varten sinun ei tarvitse toteuttaa koodia tietokantayhteyttä varten. Sen sijaan simuloimme yhdistämistä ja yhteyden katkaisemista luomalla boolean-muuttujan m_isDatabaseConnected **CMyAppApp**-luokkaan. **Connect command** -funktio vaihtaa tämän muuttujan arvoa TRUEn ja FALSEn välillä.

► **m_isDatabaseConnected-muuttujan lisääminen CMyAppApp-luokkaan.**

1. Napauta hiiren kakkospainikkeella **CMyAppApp**-kuvaketta **ClassView**-välilehdellä.
2. Valitse pikavalikosta **Add Member Variable**.
3. Kirjoita **Add Member Variable** -dialogin **Variable Type** -ruutuun **BOOL**.
4. Kirjoita **Variable Name** -muokkausruutuun **m_isDatabaseConnected**.
5. Varmista, että **Public Access** -valinta on tehty. Lisää muuttuja napauttamalla **OK**. Huomaat uuden muuttujan ilmestyvän **CMyAppApp**-luokkaan.

Huomio Kun kirjoitetaan MFC-koodia, käytetään MFC:n määrittämää BOOL-tyyppiä (kokonaisluku tyyppi) C++:n sisäisen bool-tyypin sijasta. Tämä sen takia, että yhteen sopivuus MFC-funktioiden kanssa säilyisi. Ne nimittäin käyttävät ANSI-standardin mukaista bool-tyyppiä ja käyttävät BOOL-tyyppiä boolean parametreille ja paluuarvoille.

---

6. Siirry muokkaamaan **CMyAppApp**-luokan muodostimen koodia kaksoisnapauttamalla luokan kuvaketta.
7. Aseta m_isDatabaseConnected-muuttujalle alkuarvo lisäämällä seuraava rivi muodostimen runkoon (aaltosulkujen väliin):

```
m_isDatabaseConnected = FALSE;
```

Tässä harjoituksessa teet funktion, joka käsittelee **Connect**-komennon.

► **OnDataConnect()-funktion lisääminen CMyAppApp-luokkaan**

1. Avaa ClassWizard painamalla CTRL+W MyApp-projektissa. Napauta **Message Maps** -välilehteä ja tee seuraavat toimet:
   - Valitse **Class Name** -luettelosta **CMyAppApp**.
   - Valitse **Object IDs** -luettelosta **ID_DATA_CONNECT**.
   - Valitse **Messages**-luettelosta **COMMAND**.
2. Napauta **Add Function**. **Avautuva dialogi-ikkuna ehdottaa funktiolle nimeksi OnDataConnect(). Hyväksy nimi napauttamalla OK.**
3. Napauta **Edit Code**. MyApp.cpp-tiedosto avautuu ja kohdistin on sijoitettu funktion alkuun.
4. Korvaa //TODO-kommenttirivi seuraavalla koodirivillä:

```
m_isDatabaseConnected = m_isDatabaseConnected ? FALSE : TRUE;
```

Seuraavaksi sinun täytyy lisätä käsittelijä **Upload**-komennolle. Tämän komennon tarkoitus on siirtää sovelluksessa olevat tiedot tietokantaan. Koska sovelluksen tiedot on varastoitu sovelluksen dokumenttiluokkaan, paras sijoituspaikka **OnDataUpload()**-funktiolle näyttäisi olevan paras sijoituspaikka **CMyAppDoc**-luokka. Tämän harjoituksen tarkoituksiin riittää, että funktio näyttää viesti-ikkunan.

► **OnDataUpload()-funktion lisääminen CMyAppDoc-luokkaan**

1. Toista samat vaiheet kuin edellä, mutta tällä kertaa lisää **OnDataUpload()**-funktio **CMyAppDoc**-luokkaan käsittelemään **ID_DATA_UPLOAD**-komentoa.
2. Korvaa **OnDataUpload**-funktion //TODO-rivi seuraavalla koodirivillä:

```
AfxMessageBox("Upload successfully completed");
```

3. Käännä ja käynnistä MyApp-sovellus.
4. Nyt kun käsittelijät on lisätty, uudet komennot valikoissa ja työkalupalkissa ovat käytettävissä. **Connect**-komento ei tee vielä mitään, mutta **Upload**-komento avaa sanomaikkunan.

Valikkojen ja työkalupalkkien komentojen dynaaminen päivittäminen

Vaikka olisimme tehneet täysin toimivat versiot **OnDataConnect()** ja **OnDataUpload()** -funktioista, käyttöliittymä kaipaisi silti parannuksia loogisen suunnittelun kannalta. Pohdi seuraavia asioita:

- Tietokantayhteyden nykyisestä tilasta ei ole mitään merkintää - onko yhteys tietokantaan olemassa vai ei. Näin ollen emme tiedä, mikä vaikutus **Connect**-komennon valitsemisella on.
- **Upload**-komento, jonka tehtävä on siirtää tiedot sovelluksesta tietokantaan, toiminta riippuu siitä, onko tietokantaan olemassa toimiva yhteys vai ei. Nykyisessä sovelluksessa käyttäjä voi valita **Upload**-komennon, kun tietokantayhteyttä ei ole, jolloin toiminto on tuomittu epäonnistumaan.
- Valikko- ja työkalupalkkikomentojen ulkoasua täytyy usein muuttaa niin, että käyttäjä tietää, missä tilassa sovellus on. Jos valikon tai työkalupalkin komennon tila vaihtelee päällä ja pois -tilojen välillä, komennot pitäisi merkitä niin, että komennon tila on näkyvissä. Voit järjestää komennot *valintapainike-ryhmiksi*. Valintapainikeryhmässä vain yksi vaihtoehto voi olla valittuna kerrallaan. Yhden vaihtoehdon valitseminen poistaa automaattisesti aikaisemman valinnan.
- Komennot, jotka eivät pysty sovelluksen nykyisessä tilassa suorittamaan mitään hyödyllistä toimenpidettä, pitäisi poistaa käytöstä. Esimerkiksi komennon, jonka tehtävä on siirtää tietoja tietokantaan, pitäisi olla poissa käytöstä, kun tietokantayhteyttä ei ole.
- On mahdollista muuttaa valikkokomennon tekstiä riippuen sovelluksen tilasta. Tätä mahdollisuutta tulisi käyttää varoen, sillä käyttäjää hämmentää tilanne, jossa valikon komennot muuttuvat kesken sovelluksen käyttämisen.

MFC sovellusrunko tarjoaa meille helpon keinon päivittää valikkokomentojen ja työkalupalkkien painikkeiden ulkoasua antamalla mahdollisuuden tehdä käsittelijät käyttöliittymän päivityskomennoille.

Käyttöliittymän päivityskomentojen käsittelijät

Kuten muistat MFC:n sanoman käsittelyä koskeneesta luvusta 3, sovellusrunko luo käyttöliittymän päivityssanomia, jotka ilmoittavat sovellukselle, että käyttöliittymän osien tilaa tulisi muuttaa. Kun näin tapahtuu, sanomakartan kohdeobjekteista haetaan ON_UPDATE_COMMAND_UI merkintöjä, jotka yhdistävät komentotunnisteet käsittelijäfunktioihin. Tämä prosessi suoritetaan avattaessa pikavalikkoja ja työkalurivin painikkeilla sovelluksen suorituksen aikana.

Voit luoda käyttöliittymän päivityssanomille käsittelijäfunktioita, jotka käsittelevät valikkokomentojen ja työkalupalkin painikkeiden ulkoasua. Sovellusrunko lähettää funktiolle yhden parametrin - osoittimen **CCmdUI**-objektiin. **CCmdUI**-luokka sallii sinun muokata valikkokomentoja tai työkalupalkin pai-nikkeita käyttämällä yhtä taulukossa 4.1 esitetyistä jäsenfunktioista.

Taulukko 4.1 CCmd UT -jäsenfunktiot

Funktio	Tarkoitus
Enable	Kun arvoksi asetetaan TRUE, valikkokomento tai työkalurivin painike on käytettävissä. Jos arvoksi on asetettu FALSE, komento ei ole käytettävissä. Komennot, jotka eivät ole käytettävissä näkyvät himmennettyinä.
SetCheck	Kun arvoksi asetetaan TRUE, valikkokomennon vieressä näkyy valintamerkki ja työkalupalkin painike asetetaan käytettäväksi. Kun arvoksi asetetaan FALSE, valintamerkki poistuu ja työkalurivin painikkeet eivät ole enää käytettävissä.
SetRadio	Toimii kuten **SetCheck**, paitsi että sitä käytetään käyttöliittymän kohteisiin, jotka ovat osana valintaryhmää. Muiden ryhmään kuuluvien komentojen valintoja ei poisteta, jolleivät komennot itse ylläpidä valintaryhmätoimintaa.
SetText	Asettaa käyttöliittymäkomennon tekstin. Ei vaikuta työkalurivin painikkeisiin.

Seuraavassa harjoituksessa lisäät käyttöliittymän päivityskomennon käsittelijän **CMyAppApp**-luokkaan. Käsittelijä muuttaa **Connect** ja **Upload** -komentojen ja vastaavien työkalupalkin painikkeiden tilaa.

► **OnUpdateDataConnect()-käsittelijän lisääminen CMyAppApp-luokkaan**

1. Avaa ClassWizard painamalla CTRL+W MyApp-projektissa. Napauta **Message Maps** -välilehteä ja tee seuraavat toimenpiteet:
   - Valitse **Class Name** -luettelosta **CMyAppApp**.
   - Valitse **Object IDs** -luettelosta **ID_DATA_CONNECT**.
   - Valitse **Messages** -luettelosta **UPDATE_COMMAND_UI**.
2. Napauta **Add Function**. Dialogi-ikkuna avautuu ja ehdottaa käsittelijälle nimeä **OnUpdateDataConnect()**. Hyväksy nimi napauttamalla **OK**.
3. Napauta **Edit Code**. MyApp.cpp-tiedosto avautuu ja tekstikohdistin on funktion alussa.
4. Korvaa //TODO kommenttirivi seuraavalla koodilla:

```
pCmdUI->SetCheck(m_isDatabaseConnected);
```

SetCheck()-funktiolle lähetetään boolean-tyyppinen parametri. TRUE valitsee komennon ja FALSE poistaa valinnan. Edellä oleva koodi lisää visuaalisen vihjeen m_isDatabaseConnected-muuttujan tilasta eli siitä, onko yhteys tietokantaan olemassa.

Seuraavassa harjoituksessa lisää käyttöliittymän päivityskomennon käsittelijän **Update**-komennolle.

► **OnUpdateDataUpload()-käsittelijän lisääminen CMyAppApp-luokkaan**

1. Toista edellä kuvatut vaiheet, mutta lisää tällä kerralla **OnUpdateDataUpload()**-funktio **CMyAppApp**-luokkaan käsittelemään **ID_DATA_UPLOAD**-komennon UPDATE_COMMAND_UI-sanoma.
2. Kun muokkaat **OnUpdateDataUpload()**-funktiota, korvaa // TODO kommenttirivi seuraavalla koodilla:

```
pCmdUI->Enable(m_isDatabaseConnected);
```

Enable()-funktio käyttää myöskin yhtä Boolean-parametriä. TRUE asettaa komennon saataville ja FALSE poistaa sen käytöstä. Koodi vaikuttaa niin, että **Upload**-komento asetetaan käyttöön aina, kun toimiva yhteys tietokantaan on olemassa. Yhteys on olemassa, jos CMyAppApp::m_ isDatabaseConnected-muuttujan arvo on TRUE.

3. Käännä ja käynnistä MyApp-ohjelma. Huomaa, kuinka **Connect**-valikkokomento ja vastaava työkalupalkin painike näyttävät valituilta. Huomaa myös, kuinka **Connect**-komennon valitseminen tekee **Upload**-komennon valitsemisen mahdolliseksi.

Tilariville kirjoittaminen

Joissain tapauksissa pelkkä valikko- ja työkalupalkki-komentojen käyttäminen ei ole riittävä keino sovelluksen tilasta tiedottamiseen. Ajatellaanpa esimerkiksi MyApp-esimerkkisovellusta. Tällaisenaan **Connect**-komento yksinkertaisesti mahdollistaa yhteyden ottamisen ja yhteyden purkamisen yhden tietolähteen kanssa. Entäpä jos käytössä on useita vaihtoehtoisia tietolähteitä ja sovelluksella on usein tarve vaihtaa tietolähteiden välillä?

Tässä tapauksessa voisit käyttää **OnDataConnect()**-funktiota niin, että se näyttää listan, josta käyttäjä voi valita sopivan tietolähteen. Yhteyden tilan pitää selvitä käyttöliittymässä joko niin, että tietolähteen nimi on näkyvillä tai niin, että ilmoitetaan, ettei yhteyttä ole mihinkään tietolähteeseen.

Oikea paikka tällaisen tiedon esittämiseen on sovelluksen *tilarivi*. Olet jo huomannut, kuinka MFC-sovellus käyttää tilariviä valikkokomentojen kehotteiden esittämiseen. Jos käynnistät MyApp-sovelluksesi, näet, että tilarivissä esitetään myös CAPS LOCK, NUM LOCK ja SCROLL LOCK -näppäimien tila erillisissä lokeroissa.

MFC:n tuki tilariville on kapseloitu **CStatusBar**-luokkaan. **CStatusBar** antaa sinun vapaasti muokata tilarivillä olevien lokeroiden määrää, ulkoasua ja niissä esitettäviä tekstejä.

Sovellusrunko varastoi lokeroiden tiedot taulukkoon, jossa ensimmäisenä vasemmalla oleva lokero tulee paikkaan 0. Oletuksena ensimmäinen lokero on "joustava" — se täyttää tilarivin alueen, joka ei ole muiden lokeroiden käytössä, joten muut lokerot on sijoitettu oikeaan reunaan. MFC-sovellusrunko näyttää valikko- ja työkalurivikomentojen kehotteet juuri tässä ensimmäisessä lokerossa.

Kun teet uuden tilarivin, käytät merkkijonotunnisteista (yksilöivät merkkijonotaulukon merkintöihin liittyvät resurssit) koostuvaa taulukkoa, jonka sovellusrunko kytkee vastaaviin lokeroihin. Voit käyttää joko merkkijonotunnistetta tai indeksiä viitatessasi lokeroihin.

Suositeltava tapa tilarivin lokeroissa olevan tekstin päivittämiseen on käyttää ON_UPDATE_COMMAND_UI-merkintää sanomakartassa käyttöliittymän päivityskomentojen käsittelijöiden liittämiseen lokeroiden merkkijonotunnisteisiin. Voit käyttää käsittelijän *CCmdUI*-parametrin SetText()-funktiota tekstin esittämiseen lokeroissa. Huomaa, että ClassWizard ei automaattisesti liitä lokeroiden tunnisteita käsittelijöihin, joten nämä merkinnät sanomakarttaan täytyy tehdä manuaalisesti.

Lokeroiden tekstiä on mahdollista päivittää käyttämällä **CStatusBar::SetPaneText()**-funktiota. Tässäkin tapauksessa täytyy tehdä päivityskäsittelijä. Ilman käsittelijää MFC automaattisesti poistaa lokeron käytöstä ja tyhjentää sen sisällön.

Seuraavissa harjoituksissa korvaat MyApp-sovelluksen oletuslokerot, jotka näyttävät CAPS LOCK, NUM LOCK ja SCROLL LOCK -näppäinten tilat yhdellä lokerolla, joka kertoo tietokantayhteyden tilan. Tässä lokerossa näytetään sen tietolähteen nimi, johon ollaan yhteydessä. Nimi varastoidaan sovellusluokkaan. Jos sovellus katkaisee yhteyden tietolähteeseen, näytetään teksti "Database not connected". Tässä harjoituksessa ei toteuteta valikkoa, josta tietolähteen voisi valita, vaan siinä käytetään yksinkertaisesti yhtä kovakoodattua nimeä.

Seuraavassa harjoituksessa lisäät **CMyAppApp**-luokkaan tietojäsenen, jossa säilytetään nykyisen tietolähteen nimeä.

► **m_strDSN-jäsenen lisääminen CMyAppApp-luokkaan.**

1. Napauta hiiren kakkospainikkeella **CMyAppApp**-luokan kuvaketta **ClassView**-välilehdellä.
2. Valitse pikavalikosta **Add Member Variable**.

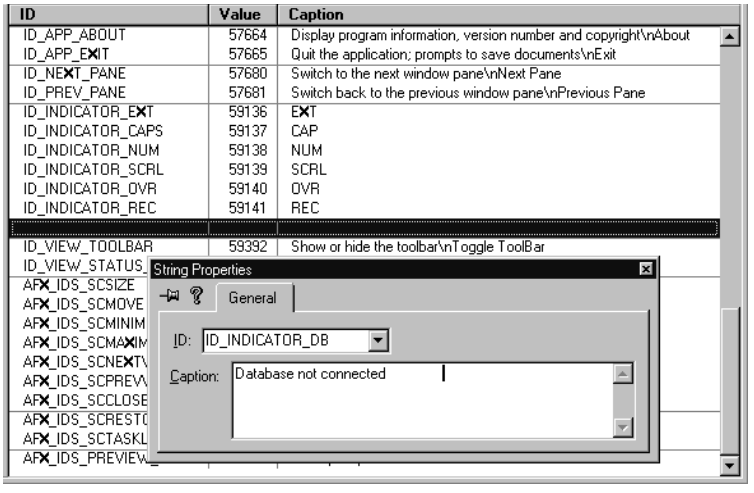
3. Kirjoita **Add Member Variable** -dialogin **Variable Type** -ruutuun **CString**.
4. Kirjoita **Variable Name** -ruutuun **m_strDSN**.
5. Varmista, että **Public Access** -vaihtoehto on valittu. Lisää muuttuja napauttamalla **OK**. Sen pitäisi ilmestyä **CMyAppApp**-luokan alle MyApp-luokkapuuhun.
6. Siirry muokkaamaan **CMyAppApp**-luokan muodostinta kaksoisnapauttamalla sen kuvaketta.
7. Lisää seuraava koodirivi muodostimen runko-osaan. Rivi alustaa m_strDSN muuttujan.

```
m_strDSN = “MyDatabase”;
```

Seuraavassa harjoituksessa luot tunnistemerkkijonon uudelle lokerolle.

► **ID_INDICATOR_DB tunnisteen lisääminen**

1. Valitse MyApp-työtilaikkunassa **ResourceView**-välilehti.
2. Avaa String Table -kansio.
3. Avaa merkkijonotaulukkoeditori kaksoisnapauttamalla **String Table** -resurssia.
4. Etsi merkkijono tunnisteet, jotka edustavat lokeroita. Jokainen näistä tunnisteista alkaa **ID_INDICATOR_**.
5. Napauta viimeistä merkintää hiiren kakkospainikkeella **ID_INDICATOR_REC**.
6. Valitse pikavalikosta **New String**.
7. Kirjoita **ID**-ruutuun **ID_INDICATOR_DB**.
8. Kirjoita **Caption**-ruutuun **Database not connected**. Tällainen oletusmerkkijono täytyy aina liittää merkkijonotaulukon merkintöihin. Lisää 15 tyhjää väliä merkkijonon loppuun välilyöntinäppäimellä. Sovellusrunko määrittää lokeron koon tämän merkkijonon perusteella ja laskee kokoon mukaan myös välilyönnit. Tyhjällä tilalla varmistetaan se, että lokeroon mahtuvat myös pisimmät merkkijonot, jotka tilassa täytyy näyttää. Näyttösi tulisi olla suunnilleen kuvan 4.10 mukainen.



Kuva 4.10 Merkkijonon lisääminen

9. Lisää uusi merkkijono painamalla ENTER. Sulje editori.

Seuraavaksi muokataan sovelluksen tilarivikoodia niin, että se luo yhden ID_INDICATOR_DB-lokeron kolmen oletuslokeron sijasta.

► MyApp-tilarivin lokeroiden asettaminen

1. Napauta MyApp-työtilaikkunassa **FileView**-välilehteä.
2. Avaa Source Files -kansio.
3. Siirry muokkaamaan **MainFrm.cpp**-tiedostoa kaksoisnapauttamalla sen kuvaketta.
4. Etsi koodista kohta, jossa tilarivin lokerotaulukko määritellään. Löydät sen välittömästi sanomakartan alapuolelta. Koodi on seuraava:

```
static UINT indicators[] =
{
    ID_SEPARATOR,         // status line indicator
    ID_INDICATOR_CAPS,
    ID_INDICATOR_NUM,
    ID_INDICATOR_SCRL,
};
```

5. Poista taulukosta kolme viimeistä kohtaa niin, että vain **ID_SEPARATOR** jää jäljelle. Lisää **ID_INDICATOR_DB**-tunniste toiseksi elementiksi. Koodin tulisi näyttää nyt seuraavalta:

```
static UINT indicators[] =
{
```

```
    ID_SEPARATOR,        // status line indicator
    ID_INDICATOR_DB,
};
```

Jos tutkit **CMainFrm::OnCreate()**-funktiota edemmäs, huomaat, että tämä taulukko annetaan parametrinä **CStatusBar::SetIndicators()**-funktiolle, jota kutsutaan **CMainFrame::m_wndStatusBar**-objektia varten.

Nyt sinun täytyy lisätä sanomakarttamerkintä ja käsittelijä käsin, jotta sovellusrunko voisi päivittää lokerot.

► **Merkinnän lisääminen sanomakarttaan**

1. Etsi sanomakartta. Sen pitäisi olla heti edellisessä harjoituksessa käsitellyn taulukon yläpuolella.
2. Lisää seuraava makro sanomakarttaan:

```
ON_UPDATE_COMMAND_UI(ID_INDICATOR_DB, OnUpdateDB)
```

Huomaa, että sanomakarttaan lisättävät merkinnät, joita ClassWizard ei ole luonut, tulisi lisätä {{AFX_MSG_MAP-kommenttien ulkopuolelle. Sanomakartan tulisi olla kokonaisuudessaan seuraavan näköinen:

```
BEGIN_MESSAGE_MAP(CMainFrame,  CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    // NOTE - ClassWizard will add and remove mapping macros here.
    // DO NOT EDIT what you see in these blocks of generated code!
    ON_WM_CREATE()
    //}}AFX_MSG_MAP
    ON_UPDATE_COMMAND_UI(ID_INDICATOR_DB, OnUpdateDB)
END_MESSAGE_MAP()
```

► **Käsittelijän lisääminen**

1. Napauta **FileView**-välilehteä MyApp-työtilassa.
2. Avaa Header Files -kansio.
3. Siirry muokkaamaan **MainFrm.h**-tiedostoa kaksoisnapautamalla sen kuvaketta.
4. Lisää seuraava funktionmäärittely **CMainFrame**-luokan määrittelyn loppuun heti **DECLARE_MESSAGE_MAP**-makron yläpuolelle:

```
afx_msg void OnUpdateDB(CCmdUI *pCmdUI);
```

   Huomaa, että myös tämä merkintä tehdään {{AFX_MSG-lohkon ulkopuolelle.

5. Siirry takaisin MainFrm.cpp-tiedostoon. Toteuta **OnUpdateDB()**-funktio lisäämällä seuraava koodi tiedoston loppuun:

```
void CMainFrame::OnUpdateDB(CCmdUI *pCmdUI)
{
    CMyAppApp * pApp = dynamic_cast<CMyAppApp *>(AfxGetApp());
    ASSERT_VALID(pApp);

    if(pApp->m_isDatabaseConnected)
        pCmdUI->SetText("Connected to: "+ pApp->m_strDSN);
    else
        pCmdUI->SetText("Database not connected");
}
```

Tämä funktio on suoraviivainen. Koodi tarkistaa tietokantayhteyden tilan tutkimalla CMyAppApp::m_isDatabaseConnected-muuttujan arvon. Jos arvo on TRUE, funktio hakee nykyisen tietolähteen nimen CMyAppApp:: m_strDSN-muuttujasta ja näyttää sen tilarivillä. Muuten se näyttää tekstin "Database not connected".

Huomaa MFC:n gloabaalin **AfxGetApp()**-funktion käyttö. Funktio palauttaa osoittimen pääsovellusobjektiin. Paluuarvo täytyy muuttaa CMyAppApp * -tyyppiseksi, jotta voit viitata CMyAppApp-jäsenmuuttujiin. Tehtäessä muutoksia perintähierarkiassa alaspäin tulisi käyttää dynamic_cast<>()-operaattoria.

Huomio Ennen kuin voit käyttää **dynamic_cast<>()**-operaattoria, sinun täytyy varmistaa, että olet valinnut **Enable Run-Time Type Information (RTTI)** -valitaruudun projektin asetuksien C/C++-sivulta kieliasetuksista. (Tarkemmat ohjeet tästä saat luvun 2 oppitunnilla 2.)

Huomaa **ASSERT_VALID** debuggausmakron käyttö osoittimen arvon tarkistamiseen.

6. Käännä ja käynnistä MyApp-ohjelma. Kokeile **Connect**-komentoa nähdäksesi, kuinka tietokantayhteyden tilasta kertova teksti tilarivillä päivittyy. Tulevissa harjoituksissa tulet muuttamaan **Connect**-komentoa niin, että yksinkertaisen päälle ja pois -valinnan sijasta avautuu dialogi, josta voit valita tietolähteen. Tässä mallissa vain tilarivi kertoo käyttäjälle, mihin tietolähteeseen (jos mihinkään) sovellus on yhteydessä.

Oppitunnin yhteenveto

MFC:n AppWizardin luoma dokumentti/näkymä-pohjainen sovellus sisältää täysin varustellun käyttöliittymän, jonka voit sovittaa vastaamaan omia vaatimuksiasi. Sovelluksen luokkien ja koodin lisäksi AppWizard luo joukon resursseja, joihin kuuluvat sovelluksen kuvakkeet, valikot ja työkalurivit, sekä merkkijonotaulukko.

Näitä resursseja voit muokata Visual C++:n kehitysympäristöön kuuluvia työkaluja käyttämällä. Grafiikkaeditoria voit käyttää kuvakkeiden ja työkalurivien painikkeiden muokkaamiseen. Menueditoria käyttämällä voit muokata valikoita visuaalisesti. Myös merkkijonotaulukon ja pikanäppäinten muokkaamista varten on olemassa editoreja.

Samalla kun käytät näitä editoreja, voit myös muokata käsittelemiesi objektien ominaisuussivuja. Näiden ominaisuussivujen avulla voit asettaa valikkokomentojen tekstejä, tilarivin kehoteteksteja ja työkalurivin työkaluvihjeitä, sekä asettaa valikoille ja työkalupainikkeille komentotunnisteita.

Komentotunnisteita käytetään valikko- ja työkalurivikomentojen ohjaamiseen oikeille käsittelijöille. ClassWizardia käyttäen voit helposti lisätä sanomakarttaan merkintöjä ja luoda käsittelijöitä mille tahansa **CCmdTarget**-luokasta periytetyille luokille. Vastaavasti voit käyttää ClassWizardia lisätäksesi käyttöliittymän päivityskomentojen käsittelijöitä. Sovellusrunko kutsuu näitä käsittelijöitä an-taakseen sinulle mahdollisuuden päivittää käyttöliittymän osien ulkoasua. Sovel-lusrunko välittää näiden funktioiden osoittimet **CCmdUI**-oliolle parametrinä. **CCmdUI**-luokka tarjoaa yhteyden käyttöliittymän elementtiin, johon käsittelijä liittyy ja sisältää jäsenfunktiot, joiden avulla voit muuttaa valikko- tai työkalurivikomentojen ulkoasua ja tilaa.

Voit myös sijoittaa tietoja sovelluksesi tilasta tilariville. MFC-luokka **CStatusBar** antaa sinulle täyden päätäntävallan tilarivin lokeroiden määrän, tyylin ja niiden sisältämien tekstien suhteen. Voit luoda taulukon tilarivin lokeroista, jotka on liitetty tunnistemerkkijonoihin sovelluksen merkkijonotaulussa. Voit käyttää näitä tunnisteita tai taulukon rivinumeroita yksittäisten tilarivin lokeroiden käsittelemiseen. Suositeltava tapa tilarivin lokeron tekstin päivittämiseen on käyttää ON_UPDATE_COMMAND_UI merkintää sanomakartassa käyttöliittymän päivityskomennon käsittelijän liittämiseen lokeron tunnisteeseen. Voit tämän jälkeen käyttää käsittelijän *CCmdUI*-parametrin **SetText()**-funktiota tekstin näyttämi-seen lokerossa. ClassWizardia ei voida käyttää lokeroiden tunnisteiden käsitteli-jöiden lisäämiseen, joten nämä merkinnät sanomakarttaan täytyy tehdä itse.

Oppitunti 2: Dialogien luominen

Tällä oppitunnilla tutustutaan Windows-sovelluksessa käytettäviin erilaisiin dialogeihin. Opit, kuinka voit visuaalisesti suunnitella ja muokata dialogeja Visual C++:n dialogieditoria käyttäen ja kuinka luot dialogin ilmentymän sovellukseesi.

Tämän oppitunnin jälkeen:

- Tiedät, minkälaisia dialogeja Windows-sovelluksessa käytetään.
- Tiedät, kuinka Visual C++:n dialogieditoria käytetään dialogimallin luomiseen.
- Tiedät, kuinka dialogin ominaisuuksia ja kontrolleja muokataan.
- Tiedät, kuinka ClassWizardin avulla luodaan mallista dialogiluokka.
- Tiedät, kuinka dialogiluokasta luodaan sovelluksessasi dialogiolio.

Oppitunnin arvioitu kesto: 30 minuuttia

Dialogit

Jokainen, joka joskus on käyttänyt Windows-sovellusta, on tutustunut dialogeihin. Dialogi on sovelluksen pääikkunan lapsi-ikkuna, jota käytetään sovelluksen tilan kertomiseen tai tietojen kysymiseen käyttäjältä. Dialogit sisältävät kontrolleja, jotka ovat pieniä standardoituja ikkunan osia, joiden avulla käyttäjä voi välittää tietoja sovellukselle tai sovellus voi välittää tietoja käyttäjälle. Esimerkiksi käyttäjä voi kirjoittaa tekstiä *muokkausruutuun* tai tehdä yksinkertaisen kyllä/ei-valinnan käyttämällä *valintaruutua*. Käyttäjä voi valita valmiiksi määrätyistä vaihtoehdoista käyttämällä luetteloruutua. Sovellus voi välittää tietoja käyttäjälle staattisen tekstikehyksen tai dynaamisen *prosenttipalkin* avulla. *Painikkeiden* avulla käynnistetään yleensä sovelluksen proseduureja tai suljetaan avoin dialogi.

Dialogityypit

On kahdenlaisia dialogeja, jotka ottavat käyttäjän syötteitä vastaan. *Modaalinen* dialogi ottaa sovelluksen käyttöliittymän haltuunsa ja vaatii käyttäjää syöttämään pyydetyt tiedot tai peruuttamaan dialogin ennen kuin sovelluksen käyttämistä voidaan jatkaa. Jokaiseen AppWizardilla luotuun ohjelmaan kuuluu **About**-dialogi, joka on hyvä esimerkki yksinkertaisesta modaalisesta dialogista. Tutustu modaalisen dialogin toimintaan valitsemalla MyApp-sovelluksen **Help**-valikosta komento **About**.

Modaalisia dialogeja käytetään kaikkialla; *modaalittomat* dialogit ovat harvinaisempia. Modaalittomat dialogit eivät ota käyttöliittymää hallintaansa, vaan antavat sinun jatkaa työskentelyä sovelluksen muissa osissa sulkematta dialogia.

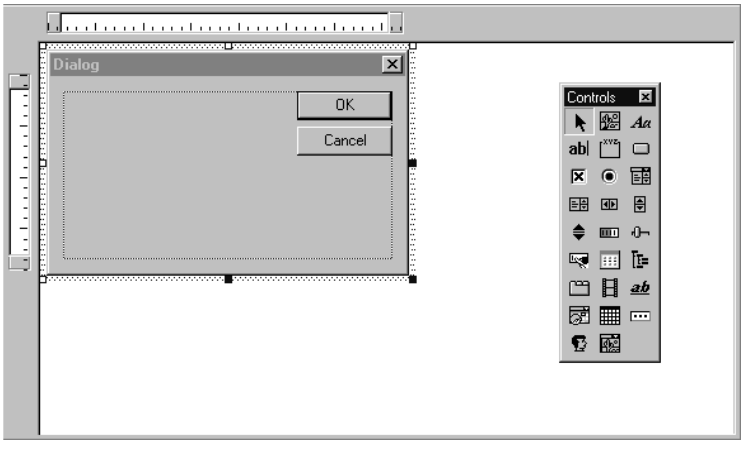
Voit siirtyä sovellusikkunoiden ja modaalittoman dialogin välillä tarpeen mukaan.

Hyvä esimerkki modaalittomasta dialogista on **Menu Item Propeties** -dialogi, jota käytit edellisellä oppitunnilla asettaessasi sovelluksen valikkokomentojen komentotunnisteita ja komentotekstejä. Avaa kokeeksi menueditori ja editoi jonkin valikkokomennon ominaisuuksia. Jos pidät **Menu Item Properties** -dialogin avoinna napauttamalla vasemmassa yläkulmassa olevaa nastakuvaketta, dialogi pysyy avoinna siirtyessäsi valikkokomennosta toiseen. Voit jopa lisätä ja poistaa valikkokomentoja Menu Item Properties -dialogin ollessa avoinna.

Menu Item Properties kuuluu itse asiassa dialogien erikoisryhmään, johon kuuluvia dialogeja kutsutaan *ominaisuusikkunoiksi*. Ominaisuusikkunaan kuuluu useita välilehdille kortiston tapaan jaettuja ominaisuussivuja, joista jokainen sisältää standardinmukaisen dialogin kontrolleja. Ominaisuusikkunoita käytetään yleensä sovelluksen objektien ominaisuuksien tai asetusten muuttamiseen.

Dialogimallin luominen

Ensimmäinen tehtävä dialogia tehtäessä on luoda *dialogimalli* Visual C++:n dialogieditoria käyttämällä. Dialogimalli on uudelleen käytettävä sovelluksen resurssi, joka sisältää dialogin ja sen sisältämien kontrollien binaarikuvaukset. Dialogieditori (kuvassa 4.11) on visuaalinen työkalu, jonka avulla voit suunnitella dialogimallin ja sen sisältämien kontrollien asettelun ruudulla, muokata niiden ominaisuuksia ja testata niiden toiminnan.



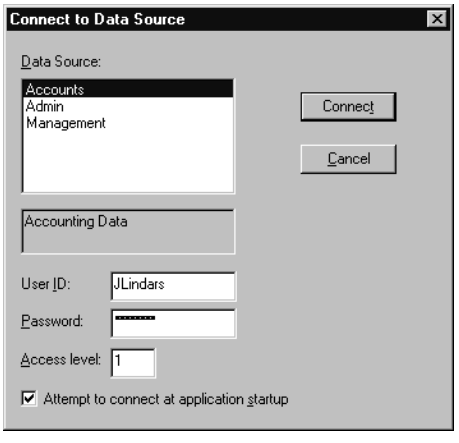
Kuva 4.11 Visual C++:n dialogieditori

Huomaa **Controls**-työkalurivi, joka voidaan myös kiinnittää paikoilleen. Kontrollien lisääminen dialogimalliin tapahtuu valitsemalla sopiva työkalu **Controls**-

työkalurivistä. Valinta muuttaa hiiren osoittimen valitun tyyppiseksi kontrollin lisäystyökaluksi (yleensä hiusristikoksi).

Seuraavissa harjoituksissa luot dialogimallin dialogille, joka kuuluu MyApp-sovellukseen. Dialogin avulla käyttäjä voi luoda yhteyden tietolähteeseen. Käyttäjä valitsee luetteloruudusta tietolähteen nimen. Vain-luku-tyyppisessä muokkausruudussa esitetään informaatiota valittuna olevasta tietolähteestä. Dialogiin asetetaan muokkausruudut, joihin käyttäjä voi syöttää kirjautumiseen tarvittavat tiedot, eli käyttäjätunnuksen, salasanan ja yhteyden tason. Valintaruutukontrollin avulla käyttäjä voi määrätä, yritetäänkö seuraavilla käyttökerroilla yhdistyä nykyiseen tietolähteeseen. Dialogissa on kaksi painiketta, **Connect** ja **Cancel**. **Connect**-painikkeella käyttäjä voi yrittää kirjautua valittuna olevaan tietolähteeseen ja **Cancel**-painikkeella toimenpiteet perutaan.

Connect to Data Source -dialogi on esitetty kuvassa 4.12. Voit käyttää tätä kuvaa mallina tehdessäsi seuraavia harjoituksia.



Kuva 4.12 Connect to Data Source -dialogi

► **Connect to Data Source -dialogimallin tekeminen**

1. Napauta MyApp työtilassa **ResourceView:**iä ja laajenna MyApp resources -kansio.
2. Napauta hiiren kakkospainikkeella Dialog-kansiota. Napauta **Insert Dialog**. Uusi tyhjä dialogi avautuu varustettuna vakiopainikkeilla **OK** ja **Cancel**.
3. Valitse dialogi napauttamalla sen otsikkopalkkia. Älä napauta kumpaakaan komentopainikkeista, sillä silloin painike tulee valituksi dialogin sijasta.
4. Muuta dialogin kooksi leveyssuunnassa 230 pikseliä ja 200 pikseliä pystysuunnassa käytäen oikeassa alareunassa olevaa koon määrityskahvaa. Dialogin koko pikseleinä näkyy dialogieditorin tilarivillä.

► **Dialogin ominaisuuksien muutaminen**

1. Paina ENTER dialogin ollessa valittuna.
2. Muuta General-välilehdellä dialogin resurssitunnistetta kirjoittamalla **ID**-ruutuun **IDD_ CONNECTDIALOG**. IDD_ on dialogien resurssitunnisteiden vakiotunniste.
3. Kirjoita **Caption**-ruutuun **Connect to Data Source**.
4. Sulje **Dialog Properties** -dialogi.

► **Otsikon lisääminen luetteloruutukontrollille**

1. Valitse **Controls**-työkaluriviltä **Static Text** -työkalu.
2. Piirrä hiusristikkokohdistimen ollessa näkyvissä tekstikehys piirto-ohjaimen oikeaan yläkulmaan.
3. Tekstikehyksen ollessa valittuna aloita **&Data Source**-tekstin kirjoittaminen; **Text Properties** -ikkuna avautuu automaattisesti ja kirjoittamasi teksti päätyy **Caption**-ruutuun.

► **Luetteloruutukontrollin lisääminen dialogiin**

1. Valitse **Controls**-työkalurivistä **List Box** -painike.
2. Piirrä hiusristikkoa käyttäen luetteloruutu juuri lisäämäsi otsikon alapuolelle. (Tarkista sopiva sijoitus kuvasta 4.12.)
3. Valitse luetteloruutu ja siirry muokkaamaan sen ominaisuuksia painamalla ENTER.
4. Kirjoita **General**-välilehden **ID**-ruutuun **IDC_DSNLIST**.
5. Tutki ominaisuusikkunan muita välilehtiä. (Painamalla F1 saat tilannekohtaisen ohjeen esille.) Kun olet valmis, sulje luetteloruudun ominaisuusikkuna.

► **Vain-luku-luetteloruudun lisääminen dialogiin**

1. Napauta **Controls**-työkalurivin **Edit Box** -työkalua.
2. Piirrä hiusristikkoa käyttäen muokkausruutu luetteloruudun alapuolelle. Tee sijoittelu kuten kuvassa 4.12.
3. Valitse muokkausruutu napauttamalla sitä ja paina ENTER.
4. Muuta kontrollin tunnistetta kirjoittamalla **General**-välilehden **ID**-ruutuun **IDC_DESCRIPTION**.
5. Valitse Styles-välilehdeltä **Multiline** ja **Read Only** -valintaruudut.
6. Sulje **Edit Properties** -dialogi.

► **Kirjautumisruutujen lisääminen dialogiin**

1. Tee otsikot muokkausruuduille kuvan 4.12 mallin mukaisesti käyttämällä **Static Text** -työkalua. Kirjoita otsikoiksi **User &ID:**, **&Password:** ja **&Access Level:**.
2. Tee muokkausruutu käyttäjätunnusta varten **Edit Box** -työkalulla (katso kuva 4.12). Aseta kontrollin tunnisteeksi **IDC_USERID**.
3. Tee muokkausruutu salasanaa varten. Aseta tunnisteeksi **IDC_PASSWORD**. Valitse ominaisuusikkunan **Styles**-välilehdeltä **Password**-valintaruutu. Tässä tyylissä käyttäjän kirjoittamat merkit korvataan näytöllä tähtimerkillä tietoturvan takaamiseksi.
4. Tee muokkausruutu **Access Level** -komennolle. Kuten kuvassa 4.12 näkyy tämän ruudun tulisi olla kapeampi kuin kaksi aikaisempaa. Aseta tunnisteeksi **IDC_ACCESS**. Valitse ominaisuusikkunan **Styles**-välilehdeltä **Number**-valintaruutu, jonka jälkeen muokkausruutu hyväksyy vain numeeriset syötteet.

► **Valintaruudun lisääminen dialogiin**

1. Valitse **Controls**-työkalurivistä **Check Box** -työkalu.
2. Piirrä muokkausruutu hiusristikkoa käyttämällä vain-luku-muokkausruudun alle kuten kuvassa 4.12. Laajenna kontrollia oikealle, kunnes kuvassa näkyvä teksti mahtuu siihen.
3. Valitse muokkausruutu napauttamalla.
4. Siirry muokkaamaan ominaisuuksia painamalla ENTER.
5. Muuta komennon tunnistetta kirjoittamalla General-välilehdellä olevaan **ID**-ruutuun **IDC_CHECKCONTROL**.
6. Kirjoita **Caption**-ruutuun **Attempt to connect at application & startup**.
7. Sulje **Edit Properties** -dialogi. Jos otsikko teksti ei näy kokonaan, käytä koonmuuttotyökalua kontrollin laajentamiseen.

► **Painikkeiden ulkoasun muuttaminen**

1. Napauta **OK**. Kirjoita **Connec&t**. **Push Button Properties** -ikkuna avautuu automaattisesti ja kirjoittamasi teksti ohjautuu **Caption**-ruutuun. Älä muuta komennon tunnistetta.
2. Sulje **Push Button Properties** -ikkuna. Järjestä **Connect**-painike ja **Cancel**-painike kuvan 4.12 mukaisesti.

Käyttäjien tulee usein voida valita arvoja dialogin kontrolleista ja tehdä muutoksia toisten kontrollien arvoihin. Käyttäjä voi esimerkiksi valita arvon luetteloruudusta tai kirjoittaa tekstiä muokkausruutuun. Yleistä on, että käyttäjän täytyy valita kahden komentopainikkeen välillä. Tämän liikkeen helpottamiseksi kontrollien tulee olla valittavissa. Kun yksittäinen kontrolli on valittuna, sanotaan, että sillä on *fokus*.

Kun kontrollilla on fokus, käyttäjä voi olla sen kanssa vuorovaikutuksessa. Kontrolli, jolla on fokus on helposti havaittavissa. Sen ulkoasu poikkeaa muista kontrolleista, koska sen ympärillä on kehys tai kohdistin on sen sisällä tai se on muulla tavoin merkitty. Käyttäjä saa näin selvän viestin siitä, mihin kontrolliin toimenpiteet vaikuttavat.

Käyttäjä voi siirtyä dialogin kontrollien välillä TAB-näppäintä käyttämällä. Kun käyttäjä painaa TAB-näppäintä, fokus siirtyy kontrollista toiseen. Dialogin kaikki kontrollit eivät välttämättä saa fokusta. Esimerkiksi tekstikehys ei saa fokusta. Sen sijaan jokainen kontrolli, joka voi fokuksen saada, saa sen vuorollaan. Järjestystä, jossa ne vuoron saavat, kutsutaan *sarkainjärjestykseksi* (tab order).

Dialogissa olevien kontrollien sarkainjärjestyksen asettaa ohjelmoija. Sarkainjärjestys antaa mahdollisuuden navigoida kontrollien välillä käyttäen pelkästään näppäimistöä. Sarkainjärjestyksellä voi olla suuri merkitys paljon käytetyissä sovelluksissa, joiden käyttäjien työteho voi laskea ja työuupumus lisääntyä, jos he joutuvat tuhansia kertoja kuukaudessa siirtämään kätensä näppäimistöltä hiirelle ja takaisin.

TAB-näppäimen painaminen siirtää fokusta eteenpäin sarkainjärjestyksessä. SHIFT+TAB-näppäinyhdistelmä siirtää fokusta takaisinpäin. Kun dialogi avautuu, ensimmäinen valittavissa oleva kontrolli saa fokuksen.

Dialogeissa, joissa on paljon kontrolleja, käyttäjä voi joutua painamaan TAB-näppäintä useita kertoja päästäkseen haluamaansa kontrolliin. Pikanäppäimet voivat olla näissä tapauksissa käyttökelpoisia. Pikanäppäimen lisäämiseksi aseta tekstikehys sarkainjärjestykseen juuri siihen liittyvän kontrollin edelle ja lisää &-merkki tekstikehyksessä näkyvään tekstiin.

Kun käyttäjä painaa pikanäppäintä, fokus siirtyy pikanäppäimeen liittyvää tekstikehystä sarkainjärjestyksessä seuraavaan kontrolliin, joka voi fokuksen saada.

► **Sarkainjärjestyksen asettaminen**

1. Valitse Visual Studion -pääikkunan **Layout**-valikosta komento **Tab Order**. Jokaisen kontrollin yläkulmaan ilmestyy numero, joka kertoo sen nykyisen paikan sarkainjärjestyksessä.
2. Napauta ensin **Data Source** -otsikkoa, sitten luetteloruutukontrollia, ja sen jälkeen jokaisen syöttöruudun otsikkoa. Valitse seuraavaksi **Attempt to connect at application startup** -valintaruutu, napauta sitten **Connect**-painiketta ja lopuksi **Cancel**-painiketta. Järjestys, jossa napautukset tehdään määrää sarkainjärjestyksen.
3. Lopeta sarkainjärjestyksen määrittäminen napauttamalla dialogin ulkopuolista aluetta.

► **Dialogin kokeileminen**

1. Napauta **Test**-painiketta tai paina CTRL+T. Dialogisi avautuu.
2. Kokeile sarkainjärjestystä. Kokeile pikanäppäimiä varmistaaksesi, että sarkainjärjestys on asetettu niidenkin osalta oikein. Valitse kokeeksi esimerkki komentoja luetteloruudusta ja yritä sitten sulkea dialogi **Continue** tai **Cancel**-painikkeella.

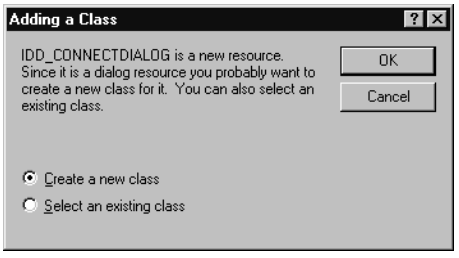
Dialogiluokan luominen ja käyttäminen

Kun olet tehnyt dialogimallin, sinun täytyy luoda seuraavaksi dialogiluokka, joka edustaa dialogia koodissasi. Dialogiluokka periytetään MFD:n luokasta **CDialog**. Tähän luokkaan voi kuulua kontrolleja ja dialogin elementtejä edustavia jäsenmuuttujia sekä metodeja, jotka käsittelevät tapahtumia, jotka syntyvät käyttäjän käsitellessä dialogin kontrolleja. Dialogiluokan luomista ja kontrollien yhdistämistä jäsenmuuttujiin helpottavat huomattavasti ClassWizardin automaattiset toiminnot.

Tällä oppitunnilla teet **Connect to Data Source** -dialogia vastaavan luokan ClassWizardia käyttäen ja opit, kuinka näytät dialogin sovelluksessasi. Luvussa 5 opit, kuinka työskentelet dialogin tietojen ja kontrollien kanssa.

► **CConflictDialog-luokan luominen**

1. **IDD_CONNECTDIALOG**-dialogin ollessa avattuna dialogieditorissa avaa ClassWizard painamalla CTRL+W. Näet kuvassa 4.13 olevan **Adding a Class** -dialogin.



Kuva 4.13 Uuden dialogiluokan lisääminen

2. Napauta **OK Create a new class** -vaihtoehdon ollessa valittuna. **New Class** -dialogi avautuu.
3. Kirjoita **Name**-ruutuun **CConnectDialog**. Huomaa tiedoston nimi, joka muodostetaan automaattisesti.
4. Napauta **OK**. **CConnectDialog** -dialogi muodostetaan ja se on valittuna ClassWizardin **Class Name** -ruudussa.
5. Sulje ClassWizard napauttamalla **OK**.

Dialogin avaaminen

Kun olet luonut dialogiluokan, modaalisen dialogin luokan tuominen näytölle tapahtuu yksinkertaisesti luomalla koodissa dialogiolio ja kutsumalla kantaluokan **CDialog::DoModal()**-metodia. Kantaluokan funktiot **CDialog:: OnOK()** ja **CDialog::OnCancel()** toimivat (oletuksena olevien) **OK** ja **CANCEL**-painikkeiden tuottamien IDOK ja IDCANCEL -sanomien käsittelijöinä. Molemmat funktiot kutsuvat **CDialog::EndDialog()**-metodia sulkeakseen dialogin. **DoModal()**-funktio palauttaa dialogin sulkemiseen käytetyn painikkeen tunnisteen.

MFC:n AppWizardilla luotu sovellus sisältää valmiiksi About-dialogin dialogimallin ja dialogiluokan. About-dialogin esittämiseen tarvittava koodi on sijoitettu sovellusobjektin **OnAppAbout()**-jäsenfunktioon, joka toimii **ID_APP_ABOUT**-komentotunnisteen käsittelijänä. Seuraava koodikatkelma näyttää, kuin-ka **OnAppAbout()**-funktiota käytetään MyApp-sovelluksessa **About**-dialogin näyttämiseen:

```
void CMyAppApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}
```

Modaalittomia dialogeja käytetään toisella tavalla. **DoModal()**-metodin kutsumisen sijasta kutsutaan **CDialog::Create()**-funktiota ja välitetään parametrinä dialogimallin resurssitunniste. Jos dialogiresurssilla on ominaisuus **WS_VISIBLE**, dialogi ilmestyy välittömästi. Jos näin ei ole, sinun täytyy kutsua kantaluokan jäsenfunktiota **CWnd::ShowWindow()**-dialogin näyttämiseksi. **ShowWindow()**-funktio vastaanottaa yhden parametrin, joka voi olla mikä tahansa valmiiksi määrätyistä arvoista, jotka kertovat, kuinka ikkuna näytetään. Arvot, joita mo-daalittoman dialogin yhteydessä yleensä käytetään, ovat SW_SHOW, jolla dia-logi tuodaan näyttöön ja SW_HIDE, jolla se piilotetaan.

Huomio Jos haluat palauttaa mieleesi **CDialog**-luokan perintähierarkian, palaa kuvaan 3.1 luvussa 3.

Koska modaaliset dialogit ottavat sovelluksen koko käyttöliittymän haltuunsa, ne luodaan yleensä vain tarvittaessa. Modaalista dialogia edustava olio luodaan juuri ennen kuin sitä käytetään ja se poistetaan, kun dialogi on suljettu ja käyttäjän antamat syötteet on käsitelty.

Modaalittomat dialogit ovat yleensä olemassa sovelluksen koko toiminnan ajan ja ne yksinkertaisesti tuodaan näkyviin tai piilotetaan käyttäjän komentojen seurauksena. Modaalittomat dialogit luodaan usein sovelluksen käynnistysvaiheessa, monesti jonkin luokan jäseninä. SW_SHOW ja SW_HIDE -arvoja käytetään dia-login esiin tuomiseen ja piilottamiseen sovelluksen tarpeiden mukaan. Modaa-littomat dialogit muistuttavat tässä suhteessa sovelluksen työkalurivejä. Työka- lurivit luodaan sovelluksen käynnistysvaiheessa osana **CMainFrame**-luokkaa ja ne ovat aina valmiina esitettäviksi tai piilotettaviksi sen mukaan kuin käyttäjä **Toolbar**-valikkokomennoilla määrää.

Opit lisää siitä, kuinka modaalisia ja modaalittomia dialogeja käytetään luvussa 5. Seuraavassa harjoituksessa opit, kuinka saat MyApp-sovelluksen **Connect**-valikkotoiminnon avaamaan **Connect to Data Source** -dialogin.

► **Connect to Data Source -dialogin avaaminen**

1. Napauta MyApp-projektissa **ClassView**-välilehteä ja laajenna **CMyAppApp**-luokan kuvake.
2. Ota esille käyttöliittymän päivityskomennon käsittelijä kaksoisnapauttamalla **OnUpdateConnect()**-funktion kuvaketta.
3. Poista koodista rivi, jossa lukee:

```
pCmdUI->SetCheck(m_isDatabaseConnected);
```

4. Etsi **OnConnect()**-käsittelijä. Korvaa seuraava valmiina oleva rivi:

```
m_isDatabaseConnected = m_isDatabaseConnected ? FALSE : TRUE;
```

riveillä:

```
CConnectDialog aCD;
aCD.DoModal();
```

5. Lisää seuraava rivi MyApp.cpp-tiedoston alkuun muiden #include-rivien joukkoon:

```
#include "ConnectDialog.h"
```

6. Käännä ja suorita MyApp-sovellus. Napauta **Data**-valikon **Connect**-komentoa ja varmista, että **Connect to Data Source** -dialogi avautuu. Huomaa, että vaikka muutimme **OK**-painikkeen caption-ominaisuuteen tekstin *Connect*, sen tunniste on edelleen **IDOK** ja näin ollen se kutsuu oletus **OnOK()**-käsittelijää suljettaessa dialogia.

Tällä hetkellä et pysty käyttämään tätä dialogia juuri mitenkään. Seuraavassa luvussa lisätään koodi, joka ottaa vastaan dialogin välittämät tiedot ja käsittelee kontrollien sanomat.

Yleiset dialogiluokat

MFC:hen kuuluu joukko **CDialog**-luokasta periytettyjä luokkia, jotka kapseloivat "yleisetdialogit", jotka ovat osa Windowsin yleiset dialogit kirjastoa COMMDLG.DLL. Nämä dialogit yksinkertaistavat yleisimpien Windowsin toimintojen toteuttamista, kuten tiedostojen hakemista ja avaamista tai tulostustyön asetusten määrittämistä. Yleisiä dialogeja tulisi käyttää aina kun mahdollista, sillä näin varmistut siitä, että sovelluksesi käyttää standardin mukaista käyttöliittymää ja täyttää Windows Logo -vaatimukset.

MFC:hen kuuluvat viisi yleistä dialogia on esitetty taulukossa 4.2.

Taulukko 4.2 Windowsin yleiset dialogit

Dialogiluokka	Kuvaus
CColorDialog	Värien valinta.
CFileDialog	Avattavan tai tallennettavan tiedoston nimeäminen.
CFindReplaceDialog	Käytetään Find (etsi) ja Replace (korvaa) toimintojen asetusten määrittämiseen.
CFontDialog	Kirjasimen valinta.
CPrintDialog	Tulostusasetusten määrittely.

Lisätietoja näistä luokista saat Visual C++:n online-ohjeen MFC-luokkadokumentaatiosta.

Oppitunnin yhteenveto

Dialogi on pääsovelluksen lapsi-ikkuna. Sitä käytetään kerrottaessa sovelluksen tilasta tai kun halutaan pyytää käyttäjältä tietoja. Dialogi sisältää kontrolleja, jotka ovat pieniä, standardin mukaisia ikkunaobjekteja, joiden avulla käyttäjä voi suorittaa toimenpiteitä tai sovellus voi esittää tietoja käyttäjälle.

Dialogeja on kahden tyyppisiä. Modaalinen dialogi vaatii käyttäjää syöttämään tietoja tai peruuttamaan dialogin ennen sovellukseen palaamista. Modaaliton dialogi ei ota sovelluksen käyttöliittymää haltuunsa, vaan antaa sinun käyttää myös sovelluksen muita osia sulkematta dialogia. Ominaisuusikkuna on dialogi-ikkunan erikoistapaus, joka koostuu erillisillä välilehdillä olevista ominaisuussivuista. Ominaisuusikkunat voivat olla modaalisia tai modaalittomia.

Ensimmäinen asia dialogia tehtäessä on luoda Visual C++:n dialogieditorilla dialogimalli. Dialogimalli on uudelleen käytettävä resurssi, joka on dialogin ja sen sisältämien kontrollien binäärinen kuvaus. Seuraavaksi täytyy periyttää MFC:n **CDialog**-luokasta luokka, joka edustaa dialogia koodissasi. ClassWizardin automaattiset toiminnot yksinkertaistavat dialogin luomista huomattavasti.

Kun olet tehnyt dialogiluokan, voit luoda koodissasi sitä vastaavan olion ja kutsua **CDialog::DoModal()**-funktiota. Jos haluat näyttää modaalittoman dialogin, kutsu **CDialog::Create()**-funktiota ja anna sille parametriksi dialogimalli. Näytä tai piilota dialogi antamalla **CWnd::ShowWindow()**-funktiolle parametriksi SW_SHOW tai SW_HIDE-tunniste.

MFC:hen kuuluu joukko **CDialog**ista periytettyjä luokkia, jotka kapseloivat Windowsin yleiset dialogit. Nämä dialogit yksinkertaistavat Windowsin yleisimpiä toimintoja ja niitä tulisi käyttää aina, jotta varmistettaisiin sovellusten yhtenäinen ulkoasu.

Laboratorio 4: STUpload-sovelluksen käyttöliittymän luominen

Tässä laboratoriossa tehdään STUpload-sovelluksen käyttöliittymä. Sovellukselle tehdään valikkokomennot, muokataan sen työkaluriviä ja tehdään dialogimallit. Oletamme, että olet suorittanut tämän luvun harjoitukset ja osaat käyttää resurssieditoreja.

Olemme tehneet sovelluksen kuvakkeet ja työkalurivin painikeet valmiiksi puolestasi. Varmista, että projektisi vastaa tätä harjoitusta korvaamalla nykyisen työskentelykansiosi sisällys kopiolla oheisrompun Chapter4\Lab\Partial \STUpload-kansiosta (ja sen alikansioista).

STUpload-sovelluksen valikkojen muokkaminen

Ensimmäisenä tässä laboratoriossa muokataan sovelluksen valikko vastaamaan laboratoriossa 1 kuvattuja asiakkaan vaatimuksia. Tässä vaiheessa kannattaa pohtia niitä toimintoja, joita sovellukseen tullaan tekemään. AppWizardin muodostamia valikoita voidaan käyttää pohjana valikoita suunniteltaessa ja tehtäessä.

Muistanet, että käyttäjä voi ladata sovellukseen ASCII-tiedostoja ja voi tallentaa tiedot sovelluksen omassa dokumenttimuodossa. Sovellukseen tehdään myös MFC-dokumentti/näkymä-sovelluksen perustulostustoiminnot. Tästä syystä **File**-valikko säytetään STUpload-sovelluksen käyttöliittymässä.

STUpload-sovellus esittää tekstitiedostosta ladatut tiedot graafisessa muodossa. Käyttäjä ei voi käsitellä tietoja, eikä kuvaajassa ole valittavia elementtejä; tästä syystä **Edit**-valikkoa ei tarvita.

► **Edit-valikon poistaminen**

1. Avaa STUpload-projekti Visual C++:ssa.
2. Valitse **ResourceView**-välilehti ja avaa STUploadin resurssit näkyviin.
3. Avaa Menu-kansio.
4. Avaa menueditori kaksoisnapauttamalla **IDR_MAINFRAME**-valikkoresurssia.
5. Napauta valikkorivin kohtaa **Edit**, paina DELETE ja vahvista **Edit**-valikon poistaminen painamalla ENTER.

Edit-valikon tilalle tehdään **Data**-valikko, jossa ovat tarvittavat toiminnot tietojen tuomiseen tekstitiedostosta, tietojen viemiseen keskustietokantaan ja keskustietokannan selaamiseen ja kyselyiden tekemiseen.

► **Data-valikon lisääminen**

1. Napauta otsikkorivin oikeassa reunassa olevaa tyhjää valikkokomentoa.
2. Raahaa tyhjä komento **File** ja **View** -valikkojen väliin.
3. Aloita tyhjän komennon ominaisuuksien muuttaminen kaksoisnapauttamalla sitä. Kirjoita **Caption**-ominaisuudeksi **&Data**.

► **Import-komennon lisääminen Data-valikkoon**

1. Napauta **Data**-valikossa olevaa tyhjää paikkaa.
2. Kirjoita merkkijono **&Import…\tCTRL+I**. Kun aloitat kirjoittamisen **Menu Item Properties** -dialogi avautuu ja kirjoittamasi teksti ohjautuu **Caption**-ruutuun.
3. Kirjoita **Prompt**-ruutuun **Import data from file\nImport from file**.
4. Napauta **Menu Item Properties** -dialogin vasemmassa yläkulmassa olevaa painiketta, jolloin se pysyy auki samalla, kun muokkaat muita valikkokomentoja.
5. Napauta toista komentoa ja sitten uudelleen **Import**-komentoa. Varmista, että editori on muodostanut komentotunnisteen **ID_DATA_IMPORT**.
6. Lisää taulukossa 4.3 luetellut komennot **Import**-komennon alle **Data**-valikkoon toistamalla äskeiset vaiheet.

Taulukko 4.3 Data-valikon komennot

Otsikko (caption)	Kehote (prompt)	ID (editorin muodostama)
&Upload\tCTRL+U	Upload data to central database\nUpload Data	ID_DATA_UPLOAD
&Query Database...\tCTRL+Q	Query the central database\nQuery Database	ID_DATA_QUERYDATABASE

Käyttäjä voi ladata sovellukseen tekstitiedoston, jossa on useampia osakkeita ja käyttäjän pitäisi pystyä valitsemaan katseltava osake. Seuraavissa harjoituksissa toteutetaan Select Fund -ikkuna — modaaliton dialogi, jossa on luettelo valittavissa olevista dialogeista. Tässä harjoituksessa tehdään valikkoon komento, jonka avulla käyttäjä voi avata ja sulkea Select Fund -ikkunan. Tämä komento sijoitetaan **View**-valikkoon, koska sen toiminto muistuttaa **View**-valikon perustoimintoja **Toolbar** ja **Status Bar**.

► **Select Fund -komennon lisääminen View-valikkoon**

1. Valitse**View**-valikon otsikko.
2. Raahaa valikon viimeisenä oleva tyhjä komento valikon yläosaan **Toolbar**-komennon yläpuolelle.
3. Kirjoita taulukon 4.4 mukaiset tiedot.

Taulukko 4.4 Select Fund -komennon tiedot

Otsikko (caption)	Kehote (prompt)	ID (editorin muodostama)
&FundSelection	View Select Fund window\nFund Selection	ID_VIEW_FUNDSELECTION

Uusien pikanäppäimien lisääminen

Seuraavaksi lisätään pikanäppäimet uusille valikkokomennoille.

► **Pikanäppäinten määrittäminen**

1. Avaa ResourceViewissä Accelerator-kansio.
2. Avaa accelerator-editori kaksoisnapauttamalla IDR_MAINFRAME-pikanäppäinresurssia.
3. Kaksoisnapauta tyhjää kohtaa pikanäppäinmerkintöjen luettelon lopussa. **Accelerator Properties** -omianisuusikkunan pitäisi avautua.
4. Valitse **ID** -luettelosta **ID_DATA_IMPORT**.
5. Kirjota **I Key**-ruutuun ja valitse **Ctrl**-valintaruutu. Poista merkintä **Shift**-valintaruudusta, jos se on valittuna.
6. Sulje ominaisuusikkuna. Pikanäppäin on lisätty taulukkoon.
7. Lisää samalla tavalla taulukossa 4.5 luetellut pikanäppäimet.

Taulukko 4.5 Pikanäppäin taulukon merkinnät

ID	Pikanäppäin
ID_DATA_UPLOAD	CTRL+U
ID_DATA_QUERYDATABASE	CTRL+Q

STUpload-sovelluksen työkalurivin muokkaaminen

Jos avaat STUpload-sovelluksen **IDR_MAINFRAME** -työkaluresurssin, huomaat, että olemme toimittaneet käyttöösi uuden työkalurivibittikartan. Bittikartta sisältää kuvakkeet **Import Upload**, ja **Query Database** -komennoille. Uusi bittikartta on kuvattu kuvassa 4.14.

Kuva 4.14 STUpload-sovelluksen työkalurivi

Seuraavaksi työkalupainikkeet täytyy liittää juuri luotujen valikkokomentojen tunnisteisiin.

► **Työkalurivin painikkeiden yhdistäminen komentojen tunnisteisiin**

1. Avaa ResourceViewissä Toolbar-kansio.
2. Avaa työkaluvieditori kaksoisnapauttamalla IDR_MAINFRAME-työkaluresurssia.
3. Napauta **Import**-työkalupainiketta.
4. Avaa **Toolbar Button Properties** -dialogi painamalla ENTER. Huomaat, että painike on edelleen yhdistetty **ID_EDIT_CUT**-tunnisteeseen.
5. Valitse **ID**-luettelosta **ID_DATA_IMPORT**. Tämä tunniste muodostettiin, kun lisäsit **Import**-komennon valikkoon.
6. Napauta nastapainiketta, jolloin **Properties**-dialogi pysyy avoimena. Muuta muiden komentopainikkeiden tunnisteet seuraamalla edellisiä ohjeita taulukon 4.6 mukaisiksi.

Taulukko 4.6 Työkalurivin painikkeiden tunnisteet

Painike	Vanha tunniste	Uusi tunniste
Upload	ID_EDIT_COPY	ID_DATA_UPLOAD
Query	ID_EDIT_PASTE	ID_DATA_QUERYDATABASE
Select Fund	ID_APP_ABOUT	ID_VIEW_FUNDSELECTION

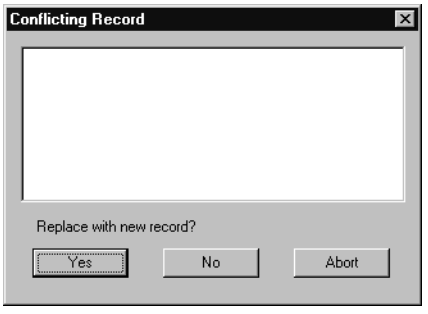
STUpload-sovelluksen dialogin lisääminen

Laboratorion tässä osassa luodaan STUpload-sovellukselle kaksi dialogia: **Conflicting Record** -dialogi ja **Select Fund** -dialogi.

Conflicting Record -dialogi

Conflicting Record -dialogi sisältää *rich edit* -kontrollin, joka on muokkauskontrolli, jossa voidaan esittää rich text format (RTF) -muotoista tekstiä ja kolme painiketta, **Yes**, **No** ja **Abort**. Dialogi varoittaa käyttäjää ristiriitaisista tiedoista tietoja tuotaessa. Muokkausruudussa esitetään niiden olemassaolevien tietueiden ja uusien tietueiden tiedot, joissa esiintyy sama osake ja päiväys, mutta eri hintaan. Käyttäjä voi päättää korvataanko olemassaoleva tieto, jätetäänkö tieto muuttamatta vai perutaanko koko tuontiprosessi.

Conflicting Record -dialogi on kuvassa 4.15. Voit käyttää kuvaa mallina seuraavissa laboratorion osissa.



Kuva 4.15 Conflicting Record -dialogi

► **Conflicting Record -dialogin luominen**

1. Avaa ResourceViewissä STUpload-resurssikansio.
2. Kaksoisnapauta Dialog-kansiota. Napauta **Insert Dialog**. Uusi tyhjä dialogi avautuu.
3. Muuta dialogin kokoa niin, että se on 7.5 cm korkea ja 10 cm leveä.
4. Aloita ominaisuuksien muokkaaminen painamalla ENTER.
5. Muuta **General**-välilehdellä dialogiresurssin tunnisteeksi **IDD_CONFLICT_ DIALOG**.
6. Kirjoita **Caption**-ruutuun **Conflicting Record**.
7. Muuta **OK**-painikkeen otsikoksi **Yes**, mutta älä muuta tunnistetta IDOK. Muuta **Cancel**-painikkeen otsikoksi **No**, mutta älä muuta komennon tunnistetta IDCANCEL.
8. Kopioi **Cancel**-painike uuden painikkeen pohjaksi. Napauta uutta painiketta ja kirjoita sen otsikoksi **Abort**. Valitse **ID**-luettelosta **IDABORT**. Sulje **Properties**-dialogi.
9. Järjestä painikkeet kuvan 4.15 mukaiseen järjestykseen.

► **Selitetekstin lisääminen dialogiin**

1. Etsi ja valitse **Static Text** -työkalu.
2. Piirrä **Yes**-painikkeen yläpuolelle selitetekstiruutu, joka on dialogin levyinen, muttei yhtä tekstiriviä korkeampi.
3. Kirjoita selitetekstin ollessa valittuna **Replace with new record?**
4. Sulje avautunut **Text Properties** -dialogi. Varmista, että teksti on oikeassa kohdassa ja näkyy kokonaan kuvan 4.15 avulla.

► **Rich edit -kontrollin lisääminen dialogiin**

1. Etsi työkaluvihjeen avulla **Rich Edit** -työkalu **Controls**-työkaluriviltä ja napauta sitä. Älä sekoita sitä tavalliseen Edit Box -työkaluun.
2. Piirrä kuvan 4.15 mukainen rich edit -kontrolli. Sen tulisi olla editorin piirto-ohjaimien levyinen ja niin korkea, että siihen mahtuu kuusi riviä tekstiä.
3. Aloita ominaisuuksien muokkaaminen painamalla ENTER.
4. Muuta **General**-välilehdellä kontrollin tunnisteeksi **IDC_CONFLICT_ RICHEDIT**.
5. Valitse **Styles**-välilehdeltä **Multiline** ja **Read Only** -valintaruudut.
6. Sulje **Rich Edit Properties** -dialogi.
7. Valitse Visual Studion pääikkunan **Layout**-valikosta **Tab Order**.
8. Aseta sarkainjärjestys napauttamalla ensin **Yes**-painiketta, sitten **No**-painiketta ja sitten **Abort**-painiketta.
9. Kokeile dialogin toiminta painamalla CTRL+T.

Select Fund -ikkuna

Kuten aiemmin kohdassa *STUpload-sovelluksen valikon muokkaaminen* todettiin, Select Fund -ikkuna tullaan toteuttamaan modaalittomana dialogina. Tämä dialogi sisältää luetteloruudun, josta käyttäjä voi valita jonkin sillä hetkellä sovellukseen ladatun osakkeen.

Select Fund -dialogi on kuvassa 4.16.

Kuva 4.16 Select Fund -dialogi

► **Select Fund -dialogin luominen**

1. Napauta ResourceViewissä hiiren kakkospainikkeella Dialog-kansiota.
2. Napauta **Insert Dialog**. Uusi tyhjä dialogi avautuu. Muuta dialogin korkeutta niin, että se on pystysuunnassa noin 4.5 cm korkea.
3. Aloita ominaisuuksien muokkaaminen painamalla ENTER.
4. Kirjoita **ID**-ruutuun **IDD_FUNDDIALOG**.
5. Kirjoita **Caption**-ruutuun **Select Fund**.
6. Poista valinta **Styles**-välilehden kohdasta **System menu**.

7. Aseta valinta **Extended Styles** -välilehden kohtaan **Tool window**.
8. Sulje **Dialog Properties** -ikkuna.

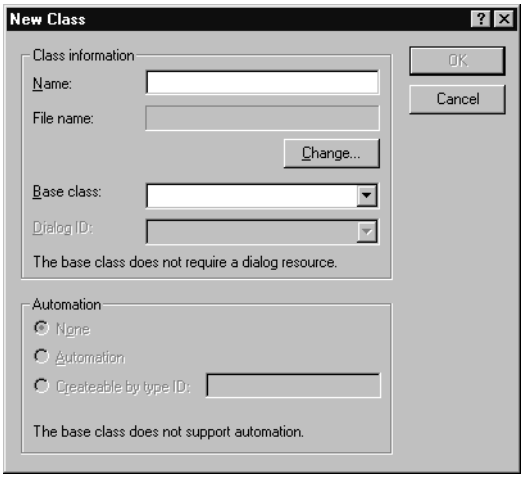
► **Luetteloruudun lisääminen**

1. Valitse **List Box** -komento **Controls**-työkaluriviltä.
2. Muuta luetteloruudun korkeutta niin, että se mahtuu piirtoalueelle. Luetteloruudun tulisi olla leveydeltään noin 2,5 cm kuten kuvassa 4.14.
3. Aloita ominaisuuksien muokkaaminen painamalla ENTER.
4. Kirjoita **General**-välilehden **ID**-ruutuun **IDC_FUNDLIST**.
5. Sulje luetteloruudun ominaisuusikkuna.

Seuraavaksi lisätään **Conflicting Record** ja **Select Fund** -dialogeja vastaavat luokat. Tämän luvun oppitunnilla 2 opit, kuinka luokka voidaan luoda automaattisesti käynnistämällä ClassWizard dialogieditorista. Seuraavassa harjoituksessa luodaan dialogille luokka käyttämällä ClassWizardia niin, että dialogieditori ei ole auki. Näin nähdään, että ClassWizard automatisoi *minkä tahansa* MFC-luokasta periytetyn luokan luomisen.

► **Dialogiluokan luominen**

1. Valitse **Window**-valikosta **Close All**.
2. Avaa ClassWizard painamalla **ClassView**-välilehdellä CTRL+W.
3. Napauta **Add Class**. Valitse ponnahdusvalikosta **New**.
4. **New Class** -dialogi avautuu kuten kuvassa 4.17.



Kuva 4.17 **New Class** -dialogi

5. Kirjoita **Name**-ruutuun **CConflictDialog**.
6. Valitse **Base Class** -luettelosta **CDialog**. Huomaa, että voit periyttää luokan mistä tahansa MFC-luokasta. Huomaa myös, että **CDialog**-luokan valitseminen tuo **Dialog ID** -luettelon käyttöön.
7. Valitse **Dialog ID** -luettelosta **IDD_CONFLICT_DIALOG**.
8. Luo uusi luokka napauttamalla **OK**.
9. Luo dialogiluokka **Select Fund** -dialogille toistamalla samat toimenpiteet. Luokan nimeksi tulee **CFundDialog**, luokka periytetään **CDialog**-luokasta, ja luokan tulee pohjautua dialogitunnisteeseen **IDD_FUNDDIALOG**.
10. Sulje ClassWizard napauttamalla **OK**. Huomaa, että uudet luokat on lisätty **MyApp classes** -puuhun.

Yleisten dialogien käyttäminen

Laboratorion viimeisessä osassa opit, kuinka Windowsin yleistä tiedostodialogia käytetään luomalla ilmentymän MFC-luokasta **CFileDialog**. STUpload-sovelluksen **Import**-komentoa käytetään sovellukseen tuotavien tekstitiedostojen paikallistamiseen ja avaamiseen. **Import**-komennolle tehdään käsittelijä, joka (toistaiseksi) avaa yksinkertaisesti Windowsin yleisen tiedostodialogin, jossa tiedostosuodattimeksi on asetettu .dat-tarkennin. Koska käsittelijän tehtävänä on ladata tietoja sovellukseen, funktio toteutetaan sovelluksen dokumenttiobjektin **CSTUploadDoc** osana.

► **Import-komennon käsittelijän luominen**

1. Avaa ClassWizard painamalla CTRL+W.
2. Valitse **Class Name** -luettelosta **CSTUpload**.
3. Valitse **Object IDs** -luettelosta **ID_DATA_IMPORT**.
4. Valitse **Messages**-luettelosta **COMMAND**.
5. Napauta **Add Function**, ja hyväksy nimi **OnDataImport** napauttamalla **OK**.

► **OnDataImport()-funktion toteutus**

1. Aloita funktion muokkaaminen napauttamalla **Edit Code**.
2. Etsi Windowsin resurssienhallinnan avulla \Chapter 4\Code-kansio oheisrompulta.
3. Avaa CH4_01.cpp-tiedosto muokattavaksi Visual C++:ssa kaksoisnapauttamalla sitä. Tiedosto sisältää seuraavan koodin:

```
// String to customize File Dialog
CString strFilter = Data Files (*.dat)|*.dat|All Files (*.*)|*.*||";
CFileDialog aFileDialog(TRUE, NULL, NULL,
      OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT, strFilter);
int nID = aFileDialog.DoModal();
```

   Tämä koodi luo MFC-luokasta **CFileDialog**-objektin. Muodostimelle välitetyssä merkkijonossa määritellään, että Windowsin yleinen tiedostovalikko luodaan niin, että siinä suodatetaan näkyviin vain .dat-päätteiset tiedostot, vaikka käyttäjä voi valita myös vaihtoehdon “all files”, eli *.*-suodattimen. Dialogi aktivoidaan kutsumalla kantametodia **CDialog::DoModal()**.

4. Valitse CH4_01.cpp-tiedoston teksti kokonaisuudessaan ja kopioi koodi leikepöydälle painamalla CTRL+C.
5. Palaa **OnDataImport()**-funktion toteutukseen. Valitse rivi // TODO kokonaisuudessaan.
6. Korvaa tämä rivi leikepöydällä olevalla koodilla painamalla CTRL+V.

Voit nyt kääntää ja käynnistää STUpload-sovelluksen. Kokeile valikon ja työkalurivin vihjeominaisuuksia. Valitse **Import**-komento **Data**-valikosta ja varmista, että tiedostodialogi avautuu odotetulla tavalla. Voit hakea Ch4Test.dat-tiedoston oheisrompun \Chapter 4\Data-kansiosta. **Import**-komennon muokkaamista jatketaan laboratoriossa 5.

Et voi tässä vaiheessa avata luomiasi dialogeja. Vaikka mallit ja luokat niitä varten on tehty, dialogien ilmentymiä ei sovelluksessa ole vielä luotu. Ne luodaan laboratoriossa 5.

Kertaus

1. Mitä kuvakkeita on toimitettava Windows 98/Windows NT logo -vaatimusten mukaisessa sovelluksessa?

2. Kuinka työkalurivin vihjetekstit toteutetaan?

3. Kuinka sovelluksen koodista voidaan muuttaa valikon tekstiä dynaamisesti?

4. Mikä on suositeltava tapa tilarivin osoittimen päivittämiseen?

5. Kuinka dialogin muokkausruudusta tehdään vain-luku-tyyppinen? Miksi näin menetellään?

6. Olet tehnyt dialogimallin dialogieditorilla. Kuinka avaat sovelluksessasi tämän mallin mukaisen modaalisen dialogin?