L U K U 1 2

Internet-ohjelmointi

Tässä luvussa

Tässä luvussa esitellään muutamia Microsoft Visual C++ 6.0:n ominaisuuksia, joiden avulla voit tehdä internet-pohjaisia sovelluksia. Opit, kuinka Dynamic HTML:ää (DHTML) käytetään sovelluksen käyttöliittymän tekemiseen ja kuinka voit lisätä selainominaisuuksia ohjelmiin ja komponentteihin. Opit, kuinka tehdään *ActiveX-dokumentteja* — dokumentteja, joita Microsoft Internet Explorer isännöi — Internetissä katseltaviksi ja muokattaviksi. Opit lisäksi, miten tehdään Internet API (ISAPI) DLL:iä, jotka lisäävät Microsoft Web-palvelimien (Internet Informatioin Server, IIS ja Personal Web Server, PWS) tarjoamia palveluja.

Ennen kuin aloitat

Ennen tämän luvun aloittamista sinun tulisi lukea luvut 2-11 ja tehdä niihin liittyvät tehtävät. Järjestelmääsi pitäisi olla asennettuna joko Internet Explorer 4.0 tai uudempi, ja IIS 4.0:n (jos käytät Microsoft Windows NT Serveriä) tai PWS:n (jos käytät Windows NT Workstationia tai Microsoft Windows 95/98:aa) tulisi olla asennettuna *Tästä kirjasta* -luvussa kerrotulla tavalla.

Oppitunti 1: Dynamic HTML

Hypertext Markup Language (HTML) on järjestelmä, jonka avulla dokumenttiin tehdään World Wide Webissä julkaisemiseen tarvittavat muotoilumerkinnät. Dynamic HTML (DHTML) on HTML:n laajennus, jota Internet Explorer 4.0 tukee. Se julkistaa kaikki Web-sivun elementit objekteina, joita voidaan käsitellä skripteissä. Tällä oppitunnilla opit, kuinka HTML-dokumentti avataan ja kuinka sitä käsitellään DHTML-objektimallin kautta Visual C++ -sovelluksissa.

Tämän oppitunnin jälkeen:

- Tiedät joitain DHTML-käyttöllittymän eduista.
- Tunnet DHTML-objektimallin ja kuinka DHTML-objekteja käsitellään ohjelmakoodista.
- Tiedät, kuinka **DHTMLView**-luokkaa käytetään HTML-dokumentin näyttämiseen MFC-sovelluksessa.
- Tiedät, kuinka ATL HTML -kontrollia käytetään HTML-dokumenttien näyttämiseen.
- Tiedät, mitä ovat DHTML-skriptletit ja kuinka niitä voidaan yhdistää Visual C++ -sovellukseen.

Oppitunnin arvioitu kesto: 50 minuuttia

Internet, intranet ja HTML

Internet on maailmanlaajuinen tietoverkko, jossa käytetään TCP/IP-protokollaa. Intranet on myös tietoverkko, jossa käytetään TCP/IP-protokollaa, mutta se ei ole maailmanlaajuinen. Yleensä intranetillä on rajoitettu määrä käyttäjiä. Yritys voi käyttää intranetiä tiedon välittämiseen omille työntekijöilleen ja jakaa tietoa suurelle yleisölle erillisen Internet-palvelun kautta. Yrityksen työntekijät voi-vat käyttää molempia, mutta ulkopuoliset voivat käyttää vain Internet-palvelua.

Huomio Tässä luvussa termiä Internet käytetään viitattaessa sekä maailmanlaajuisiin tietoverkkoihin että yritysten sisäisiin intraneteihin.

Useimmat ihmiset ovat tutustuneet Internetiin selaamalla *World Wide Webiä* (Web). Web on Internetin osa, jossa sivuja ja muita resursseja yhdistetään toisiinsa Hypertext Transfer Protocol (HTTP) -protokollaa käyttämällä. Jos olet yhteydessä Internetiin, voit käyttää toisella puolella maapalloa olevaa Web-palvelua Internet Explorerin kaltaisella Web-selaimella.

Web toimii asiakas/palvelin-periaatteella. Työasemassa sijaitseva selain toimii asiakkaana ja Web-palvelin toimii palvelimena tarjoten HTTP-sivuja, joita käyttäjä pyytää.

Selaimella katseltavat web-sivut muodostuvat tekstistä, kuvista, äänistä, digitaalisista elokuvista ja linkeistä toisille sivuille. Web sivut on tehty käyttäen HTML-kieltä, joka sisältää komennot, joilla ASCII-dokumenttiin lisätään muotoilut viittaukset grafiikkaan, ääniin, animaatiotiedostoihin ja hyperlinkit web-sivuille sekä muihin resursseihin.

Myös intranetin käyttäjät hyödyntävät web-selainta selatessaan yrityksen Web-palvelimella olevia sivuja. Lisäksi käyttäjät voivat web-selainta käyttäen hyödyntää yrityksen lähiverkossa olevia muita resursseja ja käynnistää ohjelmia etäpalvelimilta tai omassa työasemassaan. Voit käyttää HTML:ää luodaksesi sovellukseen monipuolisen ja yhtenäisen käyttöliittymän, jonka avulla voidaan muodostaa saumaton yhteys niin paikallisen työaseman, yhtiön etäpalvelimen kuin eri puolilla maailmaa olevien Web-palvelimien resursseihin.

HTML-dokumentit

HTML määrittelee joukon muotoilukomentoja, jotka kaikki Web-selaimet tunnistavat. Tässä on esimerkiksi yksinkertaisen HTML-sivun koodi:

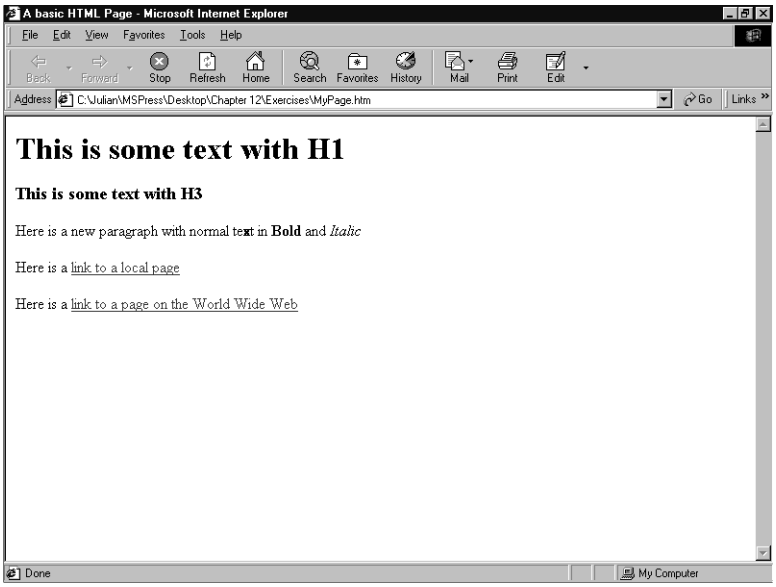
```
<!— MyPage.htm —>
<HTML>
<HEAD>
<TITLE>
A basic HTML Page
</TITLE>
</HEAD>
<BODY>
<H1> This is some text with H1
</H1>
<H3> This is some text with H3
</H3>
<P>Here is a new paragraph with normal text in
<B>Bold</B> and <I>Italic</I>
</P>
<P>Here is a <A HREF= “SecondPage.htm”>
link to a local page </A>
</P>
</P>
<P>Here is a <A HREF= “http://www.microsoft.com/visualc/”>
link to a page on the World Wide Web </A>
</P>
</BODY>
</HTML>
```

HTML-muotoilumerkinnät jakavat dokumentit erillisiin osiin, joita kutsutaan elementeiksi. Juuri nähty HTML-dokumentti koostuu kahdesta pääelementistä, jotka ovat <HEAD> ja <BODY> -merkintäparien välissä. <HEAD>-elementin sisältämää tietoa ei näytetä esitettäessä dokumenttia selaimessa, mutta selain voi hyödyntää sen sisältämää tietoa muulla tavoin. <BODY>-elementti pitää

sisällään sivusta sen tiedon, joka näytetään selaimessa. Esimerkissämme on muutamia yksinkertaisia muotoilukomentoja, jotka määrittelevät joitain dokumentin osia. Nämä komennot määräävät, näytetäänkö teksti jonkin esimääritellyn otsikkomuotoilun kuten <h1> tai <h3> mukaisesti vai lihavoituna tai kursivoituna. <A>-komento määrittelee hypertekstilinkin.

Useimpiin HTML komentoihin liittyy attribuutteja, joiden avulla annetaan elementtiin liittyviä lisämääreitä. Juuri esitetyssä esimerkissä <A>-komentoon kuuluu *HREF*-attribuutti, johon liittyvä merkkijono määrittelee linkin kohteen URL-osoitteen. HREF-attribuuttina voidaan määrittää myös paikallinen tiedosto, jolloin polkuun liitetään etuliite “file:\\”.

MyPage.htm-tiedosto on kansiossa Chapter 12\Exercises, jonne se on asennettu oheisrompulta. Avaa MyPage.htm katseltavaksi Internet Exploreriin kaksoisnapauttamalla sen kuvaketta. Lopputuloksen tulisi muistuttaa kuvaa 12.1. Voit kokeilla linkkiä SecondPage.htm-tiedostoon, ja jos olet yhteydessä Internetiin, voit testata linkin Visual C++:n kotisivulle www.microsoft.com/visualc/.



Kuva 12.1 MyPage.htm avattuna Internet Explorer 5 -selaimeen

Dynamic HTML

DHTML on kiehtova tekniikka, joka julkistettiin osana Internet Explorer 4.0:aa. DHTML:ää käyttämällä voit luoda dynaamisia vuorovaikutteisia Web-sivuja, joiden käyttöliittymä on näin ollen sivujesi käyttäjille ennestään tuttu sovellusohjelmista.

DHTML:ää käyttäen voit esimerkiksi:

- Muokata web-sivullasi olevaa tekstiä.
- Muokata tekstin ulkoasun määrittävän ominaisuussivun elementtejä (esimerkiksi fonttia ja tekstin väriä).
- Animoida sivuillasi olevia tekstejä ja kuvia.
- Vastata käyttäjän tekemiin toimintoihin kuten hiiren liikkeisiin tai napautuksiin.
- Tarkistaa käyttäjän syöttämät tiedot ennen kuin ne välitetään palvelimelle.

DHTML toteuttaa nämä toiminnot muodostamalla ja näyttämällä nykyisen dokumentin automaattisesti uudelleen sen muuttuessa. Sen ei tarvitse ladata dokumenttia uudelleen, eikä uutta dokumenttia, eikä se vaadi etäpalvelimelta sivun muodostamista uudelleen, kuten HTML vaatisi. Sen sijaan se käyttää hyväkseen käyttäjän konetta muodostaessaan ja esittäessään muuttunutta sivua. Tästä syystä käyttäjän ei tarvitse odottaa tekstin ja datan latautumista palvelimelta ennen kuin näkee tuloksen.

DHTML ei myöskään vaadi lisätukea sovelluksilta, eikä upotettuja kontrolleja muutosten näyttämiseksi. Tavallisesti DHTML-dokumenti sisältää kaikki tarvittavat tiedot, käyttää HTML-tyylejä dokumentin ulkoasun määrittelyyn ja hyödyntää pieniä skriptiosuuksia käyttäjän syöttämien tietojen käsittelemiseen ja HTML-muotoilujen, attribuuttien, tyylien ja tekstien suoraan muokkaamiseen.

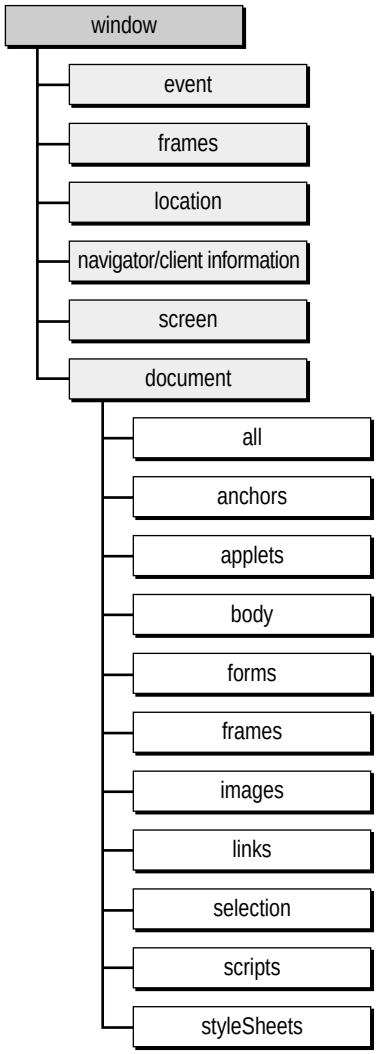
DHTML:n HTML-elementit, attribuutit ja tyylit perustuvat olemassaoleviin HTML-määrittelyihin. Microsoft työskentelee World Wide Web Consortiumissa (W3C) DHTML-standardin muodostamiseksi. Jotkin dynaamiset ja vuorovaikutteiset ominaisuudet, joita dokumentteihisi lisäät, eivät välttämättä ole täysin toimivia, jos käytetään selainta, joka ei tue DHTML:ää. Voit kuitenkin varmistaa, että dokumenttiasi voidaan katsella muillakin selaimilla, kun noudatat muutamia perusohjeita.

DHTML-objektimalli

Ladatessaan HTML-sivun Internet Explorer 4.0 (tai myöhempi versio) muodostaa siitä sisäisen mallin, johta voidaan käsitellä DHTML-objektimallin kautta.

Käyttämällä DHTML:n objektimallia voit käsitellä käytännöllisesti katsoen kaikkia dokumenttiin kuuluvia osia. Dokumentin HTML-elementtejä voidaan käsitellä yksittäisinä objekteina, eli voit tutkia ja muokata elementtiä ja sen attribuutteja lukemalla ja asettamalla sen ominaisuuksia ja kutsumalla sen metodeja. Dokumentin elementit sisältyvät *kokoelmiin* (collections) — samantapaisiin kuin MFC:n kokoelmat tai C++:n linkitetyt listat. Voit käydä DHTML-kokoelman kohta kohdalta läpi saadaksesi yhteyden sen sisältämiin elementteihin.

Kuvassa 12.2 näet joukon dokumenttiobjektin aliobjekteja. Useimmat näistä ovat kokoelmia, mukaan luettuna *all*-kokoelma, joka sisältää dokumentin HTML-elementit.



Kuva 12.2 DHTML:n objektimalli

Objektimallin avulla käyttäjän toimia, esimerkiksi näppäimen painamista tai hiiren napauttamista, voidaan käsitellä tapahtumina. Voit siepata ja prosessoida näitä ja muita tapahtumia tekemällä tapahtumankäsittelijäfunktioita ja -rutiineja.

Katsotaan esimerkiksi seuraavaa yksinkertaista DHTML-dokumenttia:

```
<!— MyDynPage.htm —>
<HTML>
<HEAD><TITLE>Welcome!</TITLE></HEAD>
<SCRIPT LANGUAGE=”JavaScript”>
function changeMe() {
      document.bgColor = “hotpink”;
      MyHeading.style.color = “green”;
      MyText.innerText = “Color effects courtesy of DHTML”;
}
</SCRIPT>
<BODY onclick=”changeMe()”>
<H3 ID=MyHeading>Welcome to Dynamic HTML!</H3>
<P ID=MyText>Click anywhere in this document.</P>
</BODY>
</HTML>
```

Tämän dokumentin taustaväri, otsikon fontti ja teksti muuttuvat käyttäjän napauttaessa dokumenttia hiirellä. Voit kokeilla tätä sivua (Chapter 12\Exercises\MyDynPage.htm) Internet Explorerilla.

Huomaa, että voit yksilöidä tietyn sivulla olevan elementin asettamalla ID-attribuutin elementin muotoilukomennossa. Esimerkissä otsikko on nimetty tunnisteella *MyHeading* ja tekstikappale tunnisteella *MyText*. Voit käyttää näitä tunnisteita viitatessasi sivun elementteihin kirjoittaessasi skriptejä, jotka käsittelevät elementtien ominaisuuksia. Skirptit määritellään <SCRIPT>-merkinnöillä.

Esimerkistä kannataa huomata, että dokumentin <BODY>-merkinnässä **onClick**-tapahtumaan yhdistetyn **changeMe()**-funktion määrittelyssä on käytetty JavaScript-kieltä. Funktio muuttaa dokumentin taustavärin asettamalla **document**-objektin bgColor-arvon. Funktio muuttaa myös otsikon värin asettamalla **MyHeading**-elementin **style**-objektin **color**-ominaisuuden. Teksti muutetaan asettamalla **MyText**-elementin **innerText**-ominaisuudelle uusi arvo.

Huomio Tämän luvun skripti esimerkeissä käytetään JavaScipt-kieltä. Se on hyvä valinta skriptikieleksi, koska sen syntaksi on hyvin samankaltainen kuin C++:ssa. JavaScript on useimpien selainten oletuskieli, mutta on silti hyvä määrittää kieli käyttämällä <SCRIPT>-merkinnän LANGUAGE-määrettä. MSDN-dokumentaatiossa ja muissa kirjoissa saatetaan käyttää Jscriptiä. Jscript on Microsoftin versio JavaSciptistä ja se sisältää muutamia Internet Explorerissa toimivia laajennuksia.

Tässä kirjassa annetaan vain lyhyt johdatus DHTML-kieleen. Jos haluat lisää tietoja DHTML:stä, valittavissa on useita aiheesta kirjoitettuja kirjoja, kuten Scott Isaacsin *Inside Dynamic HTML* (Microsoft Press, 1997). Tämän lyhyen DHTML:n esittelyn tarkoituksena on osoittaa, että HTML — hyperteksti-

dokumenttien tekemiseen tarkoitettu yksinkertainen merkintäkieli— on kehittymässä tehokkaaksi työkaluksi, jonka avulla voidaan luoda sovelluksiin dynaaminen ja vuorovaikutteinen käyttöliittymä.

Microsoft WebBrowser -kontrolli

Helpottaakseen selainelementtien lisäämistä sovelluksiin Microsoft on tuottanut WebBrowser ActiveX-kontrollin. Tämä kontrolli on Internet Explorerin osa, jossa on Internet Explorerin pääikkunan toteutus. WebBrowser-kontrolli mahdollistaa selailun hyperlinkkejä ja URL-osoitteita käyttäen, sekä ylläpitää sivuhistoriaa, jonka kautta käyttäjä voi siirtyä aiemmin selatuille sivuille ja uudelleen eteen päin. Kontrolli käsittelee myös “suosikki”-sivujen listaa. WebBrowser-kontrolli on itse ActiveX-kontrollisäilö, joka voi isännöidä mitä tahansa ActiveX-kontrollia.

Seuraavassa osassa kerrotaan kuinka MFC:n **CHtmlView**-luokkaa (joka kapseloi WebBrowser-kontrollin) käyttäen luodaan selain-tyylinen sovellus, joka käyttää HTML:ää pääkäyttöliittymänään. Opit kuinka ATL:ää käytetään luotaessa ActiveX-kontrollia, joka isännöi WebBrowser-kontrollia.

Opit myös, kuinka DHTML-objektimallia käsitellään C++-koodista ja kuinka sitä käytetään sovelluksessa tai kontrollissa näkyvän HTML-dokumentin elementtien muokkaamiseen.

Selain-tyyliset MFC-sovellukset

Microsoft WebBrowser-kontrolli on standardin mukainen ActiveX-kontrolli, eli voit sijoittaa kontrollin MFC-projektiisi käyttämällä Components and Controls Gallerya. Kun kontrolli lisätään, luodaan **CWebBrowser2**-luokka, joka käärii **IWebBrowser2 Automation**-käyttöliittymän, jonka WebBrowser-kontrolli julkistaa. Voit käyttää **CWebBrowser2**-luokkaa hakiessasi ja asettaessasi ominaisuuksien arvoja ja kutusessasi metodeja C++:ssa Automation-tyyppien sijasta.

Suositeltavampi tapa selaintyylisen sovelluksen luomiseen on kuitenkin käyttää MFC AppWizardia ja määritellä **CHtmlView**-näkymä luokaksi. **CHtmlView**-luokka on periytetty **CView**:stä ja sisältää WebBrowser-kontrollin toiminnallisuuden MFC:n dokumentti/näkymäarkkitehtuurissa. **CHtmlView** sisältää tarvittavat jäsenfunktiot, joiden avulla voit käsitellä selaimen toiminnallisuutta koodistasi.

Huomaa, että Internet Explorer 4.0 (tai uudempi) täytyy olla asennettuna ennen kuin **CHtmlView**-luokkaa tai WebBrowser-kontrollia voidaan käyttää.

Seuraavassa harjoituksessa tehdään selaintyylinen sovellus MFC AppWizardilla.

► **MyHtmlApp-sovelluksen luominen**

1. Luo uusi **MyHtmlApp**-niminen **MFC AppWizard (exe)** -projekti napauttamalla **File**-valikosta **New**.
2. Valitse MFC AppWizardin ensimmäisessä vaiheessa **Single document**.
3. Vaiheissa 2-5 hyväksytään oletusasetukset.
4. Vaiheessa 6 napauta **Base class** -ruudusta **CHtmlView**. Luo projekti napauttamalla **Finish** ja sitten **OK**.

Kun projekti on luotu, avaa **CMyHtmlAppView**-luokka ClassViewissä. Ota **OnInitialUpdate**-funktion koodi esiin kaksoisnapauttamalla. Koodin tulisi olla seuraavanlainen:

```
void CMyHtmlAppView::OnInitialUpdate()
{
    CHtmlView::OnInitialUpdate();

    // TODO: This code navigates to a popular spot on the Web.
    // Change the code to go where you'd like.
    Navigate2(_T("http://www.microsoft.com/visualc/"),NULL,NULL);
}
```

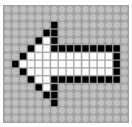
OnInitialUpdate-funktio käyttää **CHtmlView::Navigate2()**-funktiota Microsoft's Visual C++ -sivun etsimiseen ja näyttämiseen. Jos sinulla ei ole jatkuvaa Internet-yhteyttä, kannattaa todennäköisesti muuttaa **Navigate2()**-funktion kutsua niin, että avataan MyPage.htm-tiedosto. Olettaen, että MyPage.htm-tiedosto on c:\DAVC\Chapter 12\Exercises-kansiossa, rivi tulisi muuttaa seuraavaksi:

```
Navigate2(_T("c:\\DAVC\\Chapter 12\\Exercises\\MyPage.htm") ,NULL,NULL);
```

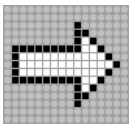
Tee seuraava harjoitus, jossa lisätään **Forward** ja **Back** -painikkeet, ennen MyHtmlApp-sovelluksen kääntämistä ja käynnistämistä.

► **Navigointi valikon ja työkalurivin toimintojen lisääminen**

1. Avaa ResourceViewissä **IDR_MAINFRAME**-valikkoresurssi **Menu**-kansiosta.
2. Poista **Edit**-valikko, sijoita sen paikalle uusi valikko **&Go**.
3. Lisää valikkoon kaksi toimintoja **&Back** ja **&Forward**. Lisää toiminnoille sopivat kehotetekstit. Anna editorin käyttää oletuksia IDs **ID_GO_BACK** ja **ID_GO_FORWARD**.
4. Sulje menueditori. Avaa **IDR_MAINFRAME** työkaluriviresurssi. Poista **Cut**, **Copy** ja **Paste** -painikkeet raahammalla ne pois työkaluriviltä.
5. Tee kuvien 12.3 ja 12.4 mukaiset painikkeet **ID_GO_BACK** ja **ID_GO_FORWARD** komennoille.



Kuva 12.3 ID_GO_BACK-painike



Kuva 12.4 ID_GO_FORWARD-painike

6. Sulje työkaluvieditori. Avaa ClassWizard ja napauta **Message Maps** -välilehteä.
7. Lisää **CMyHtmlAppView**-luokkaan komentotunnisteita **ID_GO_ BACK** ja **ID_GO_FORWARD** vastaavat käsittelijät **OnGoBack()** ja **OnGoForward()**. Etsi funktioiden toteutus napauttamalla **Edit Code**.
8. Lisää kummallekin käsittelijälle vastaava **CHtmlView**-jäsenfunktio navigointia varten seuraavaan tapaan:

```
void CMyHtmlAppView::OnGoBack()
{
    GoBack();
}

void CMyHtmlAppView::OnGoForward()
{
    GoForward();
}
```

9. Käännä ja käynnistä sovellus. Kuvassa 12.5 seuraavalla sivulla näet MyHtmlApp-sovelluksen, johon on avattu Microsoft Visual C++:n kotisivu **CMyHtmlAppView::OnInitialUpdate()**-funktion nykyisen määrittelyn mukaisesti.

Kuva 12.5 MyHtmlApp-sovellus

Kokeile CHtmlView:n sovellukseesi lisäämiä selainominaisuuksia. Huomaa, että sovelluksen tilarivi näyttää tietoja hyperlinkeistä ja sivujen lataamisen edistymisestä. Kun olet siirtynyt hyperlinkkien avulla sivulta toiselle, voit liikkua sivuhistoriassa käyttämällä juuri tekemiäsi **Go Back-** ja **Go Forward** valikko- tai työkalurivikomentoja.

DHTML-objektimallin käsittely

Muistanet, että voit helposti käsitellä DHTML-objekteja kuten **window** ja **document** kirjoittaessasi skriptiä HTML-dokumenttiin. Objektiin voidaan viitata yksinkertaisesti käyttämällä sen nimeä, kuten seuraava JavaScript katkelma osoittaa:

```
document.bgColor = “hotpink”;
```

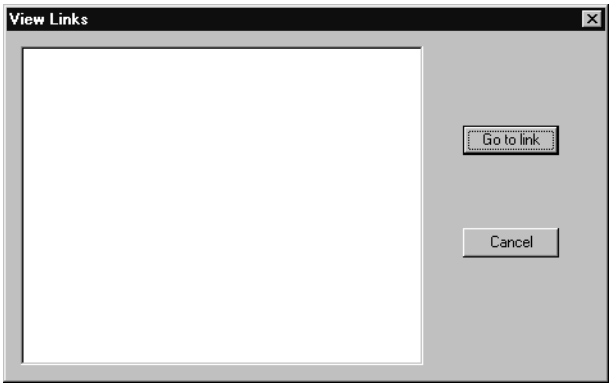
Tällaisen toiminnan mahdollistaa se, että Internet Explorerin skriptitulkki tulkitsee komentosi ja muuttaa ne kutsuiksi Internet Explorerin Automation-rajapinnalle. C++:ssa DHTML-objektimallia käytetään suoraan Automation-rajapinnan kautta.

DHTML-objektimallia käytetään muutamien etuliiteellä IHTML (**IHTMLDocument**, **IHTMLWindow**, **IHTMLElement** ja niin edelleen) varustettujen COM-rajapintojen kautta. **CHtmlView**-luokka sisältää **GetHtmlDocument()**-funktion, joka palauttaa näkyvissä olevan HTML-dokumentin **IDispatch**-osoittimen. Voit kutsua **QueryInterface()**-funktiota **IDispatch**-osoittimella saadaksesi osoittimia IHTML-rajapintoihin. Sen jälkeen, kun olet saanut osoittimet näihin rajapintoihin, voit käyttää niiden sisältämiä ominaisuuksia ja metodeja dokumenttien käsittelyyn.

Seuraavassa harjoituksessa havainnollistetaan tapaa, jolla voit käsitellä MyHtml-sovelluksen näyttämien HTML-sivujen elementtejä. Kirjoitat koodin, joka hakee kaikki ankkurielementit (linkit, links) nykyiseltä sivulta ja näyttää niiden HREF-attribuuttien arvot (URL:t tai tiedostopolut) dialogissa. Käyttäjä voi valita linkin katseltavaksi selaimessa.

► View Links -dialogin luominen

1. Luo dialogieditoria käyttäen kuvan 12.6 mukainen **View Links** -dialogi. Tälle dialogille tulisi antaa resurssitunniste **IDD_LINK_DIALOG** ja sen pitäisi sisältää suuri **IDC_LINK_LIST**-niminen luetteloruutukontrolli samoin kuin normaalit **OK** ja **Cancel** -painikkeet. **OK**-painikkeen otsikon tulisi olla **Go to link**.



Kuva 12.6 View Links -dialogi

2. Luo resurssieditorissa **View Links** -dialogille luokka **CLinkDialog** painamalla CTRL+W. Lisää ClassWizardin **Member Variables** -sivulla taulukossa 12.1 luetellut muuttujat.

Taulukko 12.1 View Links -dialogin muuttujat

Resurssitunniste	Kategoria	Tietotyyppi	Muuttujan nimi
IDC_LINK_LIST	Value	CString	m_strLink
IDC_LINK_LIST	Control	CListBox	m_lbLinkList

3. Ylikuormita **CLinkDialog**-luokan **OnInitDialog()**-funktio (WM_INITDIALOG-sanoman käsittelemiseksi) ClassWizardin **Message Maps**-välilehdellä. Etsi funktion toteutus napauttamalla **Edit Code**.
4. Korvaa funktion kanta seuraavalla koodilla, joka on tiedostossa InitLnkDlg.cpp kansiossa Chapter 12\Exercises.

```
BOOL CLinkDialog::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Get pointer to view
    CFrameWnd * pFrame =
        dynamic_cast<CFrameWnd *>(AfxGetMainWnd());
    ASSERT_VALID(pFrame);

    CHtmlView * pHtmlView =
        dynamic_cast<CHtmlView *>(pFrame->GetActiveView());
    ASSERT_VALID(pHtmlView);

    // Get pointer to the document object dispatch interface
    IDispatch * pDisp = pHtmlView->GetHtmlDocument();

    if(pDisp != NULL)
    {
        // Get pointer to IHTMLDocument2 interface to
        // access document object's properties and methods
        IHTMLDocument2 * pHTMLDocument2;
        HRESULT hr;

        hr = pDisp->QueryInterface(IID_IHTMLDocument2,
            (void**)&pHTMLDocument2);

        if(hr == S_OK)
        {
            // Get pointer to the anchors collection
            IHTMLElementCollection * pColl = NULL;
            hr = pHTMLDocument2->get_anchors(&pColl);

            if(hr == S_OK && pColl != NULL)
            {
                LONG nElem;
                hr = pColl->get_length(&nElem);

                if(hr == S_OK)
                {
```

```
                        // Iterate across anchors collection
                        for(long i = 0; i < nElem; i++)
                        {
                            _variant_t vIndex(i);
                            _variant_t vName = vIndex;

                            IDispatch * pDisp2;

                            hr = pColl->item(
                                vName, vIndex, &pDisp2);
                            if(hr == S_OK)
                            {
                                // Retrieve pointer to each
                                // Anchor element so that you can
                                // retrieve the URL text and add
                                // it to the list box
                                IHTMLAnchorElement * pAnchElem;

                                hr = pDisp2->QueryInterface(
                                    IID_IHTMLAnchorElement,
                                    (void**) &pAnchElem);
                                if(hr == S_OK)
                                {
                                    BSTR bstrHref = 0;
                                    pAnchElem->get_href(&bstrHref);

                                    CString strLink(bstrHref);

                                    if(!strLink.IsEmpty())
                                        m_lbLinkList.AddString(strLink);

                                    SysFreeString(bstrHref);
                                    pAnchElem->Release();
                                }
                                pDisp2->Release();
                            }
                        }
                    }
                    pColl->Release();
                }
                pHTMLDocument2->Release();
            }
            pDisp->Release();
        }
        return TRUE; // return TRUE unless you set the focus to a control
            // EXCEPTION: OCX Property Pages should return FALSE
}
```

Tutki tätä funktiota ja varmista, että ymmärrät, kuinka se toimii. Osoitin HTML-dokumentin saantifunkitoon saadaan kutsumalla **CHtmlView::GetHtmlDocument()**-funktiota. Tämän osoittimen avulla hankitaan **IHTMLDocument2**-rajapinnan osoitin ja kutsumalla **IHTMLDocument2:: get_anchors()**-funktiota saadaan osoitin DHTML-linkkikokoelmaan. **InitDialog()**-funktio käy läpi *anchors*-kokoelman ja jokaisella läpikäynnillä se saa **IHTMLAnchorElement**-osoittimen nykyiseen elementtiin. Tästä elementistä saadaan HREF-attribuutin arvo, joka lisätään luetteloruutuun.

5. Lisää LinkDialog.cpp-tiedoston alkuun muiden #include-komentojen joukkoon seuraavat rivit:

```
#include <mshtml.h>
#include <comdef.h>
```

mshtml.h-tiedosto sisältää IHTML-rajapintojen määrittelyt. Comdef.h sisältää **_variant_t** COM-apuluokan määrittelyn.

6. Lisää uusi valikkokomento **Links** valikkoon **View**. Sillä tulisi olla oletuskomento ID **ID_VIEW_LINKS**. Luo ClassWizardilla käsittelijä tälle komennolle **CMyHtmlAppView**-luokkaan. Funktion nimen tulisi olla nimeltään **OnViewLinks**.

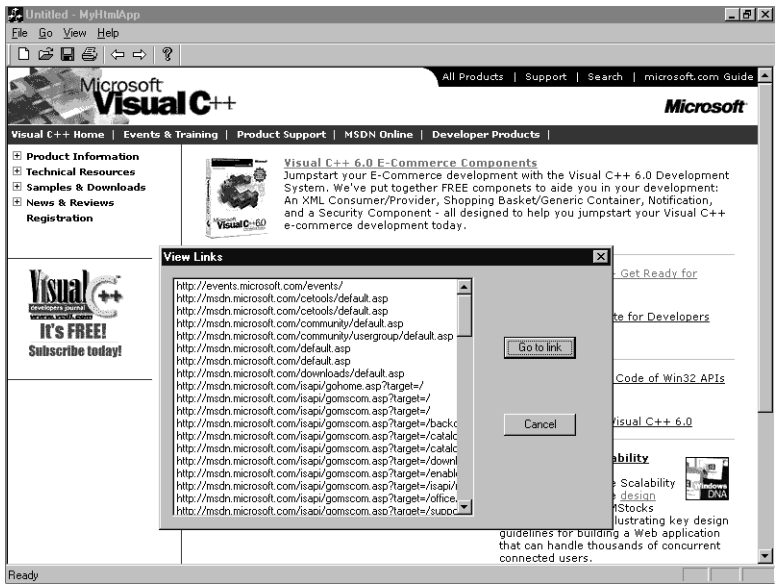
7. Lisää **CMyHtmlAppView::OnViewLinks()**-funktion runkoon seuraavat koodirivit:

```
CLinkDialog aDlg;
if(aDlg.DoModal() == IDOK)   // (“Go to link” button)
{
      // If a link was selected, go there!
      if(!aDlg.m_strLink.IsEmpty())
            Navigate2(aDlg.m_strLink, NULL, NULL);
}
```

8. Lisää MyHtmlAppView.cpp-tiedoston alkuun muiden #include-komentojen joukkoon seuraava rivi:

```
#include “LinkDialog.h”
```

9. Voit nyt kääntää ja käynnistää sovelluksen. Varmista Internetiä tai paikallisia tiedostoja käyttämällä, että kaikki kyseisen sivun linkit näkyvät dialogissa, kun **Links**-toiminto valitaan **View**-valikosta. Valitse linkki luettelosta ja varmista napautamalla **Go to link**, että haluttu kohde avautuu. Kuvassa 12.7 nähdään **View Links** -dialogi, jossa on lueteltu Microsoft Visual C++ kotisivulla olevat URL-osoitteet.



Kuva 12.7 **View Links** -dialogi toiminnassa

Huomio Välilyöntejä sisältävät URL:t, kuten c:\DAVC\Chapter 12\Exercises\MyPage.htm näkyvät **View Links** -dialogissa niin, että välilyönnit on korvattu koodinvaihtomerkeillä. Tämä johtuu siitä, että Internet Explorer käyttää **InternetCanonicalizeUrl()** API-funktiota muuntaessaan URL:iä turvalliseen muotoon. Voit käyttää **InternetCanonicalizeUrl()**-funktiota muuntaessasi URL:iä näyttämistä varten. Lisätietoja saat Internet Explorer Platform SDK:sta.

HTML-resurssit

Voit liittää HTML-sivuja projektisi resursseihin aivan samoin kuin bittikarttoja ja dialogimalleja. HTML-resurssit voidaan tallettaa .htm-tiedostoina projektin alikansioon, tai ne voidaan määritellä projektin resurssiskriptitiedostossa (.rc). Molemmissa tapauksissa voit muokata HTML-resursseja Visual C++ -editorilla. Koska HTML-resurssit käännetään osaksi ohjelmaasi, ne ovat turvallinen ja helppo tapa toimittaa sovelluksen käyttämiä HTML-sivuja.

Seuraavassa harjoituksessa tehdään MyPage.htm ja SecondPage.htm -tiedostot, jotka näit aiemmin tässä luvussa sovelluksesi resursseina.

► **HTML-resurssin lisääminen MyHtmlApp-projektiin**

1. Napauta ResourceViewissä hiiren oikealla painikkeella **MyHtmlApp Resources** -kansion ylätasoa. Napauta pikavalikosta **Import**.
2. Käyttäen **Import Resource** -dialogia, napauta **Files of type** -ruudussa **HTML files (.htm; .html)**. Etsi Chapter 12\Exercises\MyPage.htm-tiedosto ja napauta **Import**.
3. HTML-lähde avautuu editori-ikkunaan. Huomaa HTML:n mukainen värikoodaus.
4. Napauta hiiren oikealla painikkeella lisättyä **IDR_HTML1**-resurssia ja sen jälkeen **Properties**.
5. Nimeä uudelleen resurssi **IDR_MYPAGE**. Huomaa **External file** -valintaruutu, jolla määritetään, onko resurssi tallennettu projektin resurssikansioon vai .rc-tiedostoon. Anna tämän valitaruudun jäädä valituksi.
6. Lisää Chapter 12\Exercises\SecondPage.htm -tiedosto sovelluksen resurssiksi toistamalla samat toimenpiteet. Anna resurssin nimeksi **IDR_SECONDPAGE**.

Saadaksesi sovelluksen näyttämään HTML-resurssin, käytä **CHtmlView:: LoadFromResource()**-funktiota seuraavassa harjoituksessa esitetyllä tavalla.

► **HTML-resurssin näyttäminen**

1. Etsi **CMyHtmlAppView::OnInitialUpdate()**-funktio. Muuta seuraava rivi kommentiksi:

```
Navigate2(_T("http://www.microsoft.com/visualc/"),NULL,NULL);
```

Lisää seuraavaksi riviksi:

```
LoadFromResource(IDR_TITLEPAGE);
```

2. Käännä ja käynnistä MyHtmlApp-sovellus. Varmista, että **IDR_TITLEPAGE** HTML-resurssi näkyy odotetulla tavalla.

Huomaa, että näet TitlePage.htm-tiedoston käännetyn version. Tästä syystä muutokset, joita teet TitlePage.htm-tiedoston HTML-koodiin, eivät näy, jos et käännä sovellusta uudelleen. Samasta syystä **View Links**-dialogi ei pysty hakemaan eikä näyttämään tällä sivulla olevia linkkejä.

Lisäksi huomaat, että et pysty käyttämään *local page* -linkkiä. Tämä on seurausta siitä, että linkki on määritelty osoittamaan ulkoiseen .htm-tiedostoon seuraavalla tavalla:

```
<A HREF= “SecondPage.htm”> link to a local page </A>
```

Linkki täytyy muuttaa osoittamaan **IDR_SECONDPAGE**-resursiin. Internet Explorerissa on *res:*-protokolla, joka vastaa HTTP-protokollaa. Sen avulla voit määritellä URL-osoitteen, joka on sidottu suoritettavaan tiedostoon tai DLL-tiedostoon. res:-protokollan formaatti on seuraava:

```
res://resurssi_tiedosto/[resurssi_tyyppi]/resurssi_id
```

Protokollan *resurssi_tiedosto* ilmoittaa resurssin sisältävän tiedoston nimen, ja *resurssi_tyyppi* on valinnainen numero arvo, joka määrittelee resurssin tyypin — kaksi yleisimmin käytettyä arvoa ovat 23 (HTML-sivu) js 2 (bittikartta). Resurssin numeerinen tunniste määritellään antamalla *resurssi_id*. Tiedostossa winuser.h Microsoft Visual Studio\Vc98\Include-kansiossa on resurssityyppien täydellinen luettelo, jossa ne on määritelty RT_-alkuisina vakioina. Jos tyyppiä ei määritellä, käytetään oletuksena arvoa 23.

Seuraavassa harjoituksessa muutetaan HTML-sivulla olevat linkit muotoon, jossa käytetään res:-protokollaa.

► **Linkin tekeminen toiseen resurssiin**

1. Napauta ResourceViewissä hiiren kakkospainikkeella **MyHtmlApp resources** -kansiota. Napauta pikavalikosta **Resource Symbols**. **Resource Symbols** -dialogi avautuu.
2. Kirjoita muistiin **IDR_MYPAGE** ja **IDR_SECONDPAGE** -resurssitunnisteiden numeroarvot. Sulje **Resource Symbols** -dialogi.
3. Kaksoisnapauta **IDR_MYPAGE**-resurssia, jolloin pääset muokkaamaan HTML-lähdettä. Korvaa seuraava hyperlinkin määrittely:

   ```
   <A HREF= “SecondPage.htm”> link to a local page </A>
   ```

   uudella määrittelyllä:

   ```
   <A HREF= “res://MyHtmlApp.exe/n”>
   ```

   missä *n* on **IDR_SECONDPAGE**-resurssitunnisteeseen liittyvä numeroarvo.
4. Muokkaa **IDR_SECONDPAGE**-resurssia. Korvaa seuraava hyperlinkin määrittely:

   ```
   <A HREF=”MyPage.htm”> Here </A>
   ```

   uudella määrittelyllä:

   ```
   <A HREF=”res://MyHtmlApp.exe/n”> Here </A>
   ```

missä *n* on **IDR_MYPAGE**-resurssitunnisteeseen liittyvä numeroarvo.

5. Käännä ja käynnistä MyHtmlApp-sovellus. Varmista, että voit liikkua linkkien avulla MyPage ja SecondPage -sivujen välillä.

HTML-kontrollien luominen ATL:llä

Voit ATL:ää käyttämällä tehdä kontrollin, joka pystyy näyttämään HTML-sivuja. ATL *HTML-kontrolli* isännöi WebBrowser-kontrollia ja sisältää osoittimen **IWebBrowser2 Automation**-rajapintaan, jonka avulla päästään käsiksi DHTML-objektimalliin.

Kuten kaikki ATL:llä luodut kontrollit, HTML-kontrolli sisältää tyhjän rajapinnan, johon voit lisätä säiliöliittymän määrittelyyn tarvittavat metodit ja ominaisuudet. Lisäksi HTML-kontrollilla on toinen tyhjä rajapinta, jota käytetään kommunikointiin C++-koodin ja HTML-käyttöliittymän kanssa. HTML-käyttöliittymä kutsuu C++-koodia käyttämälllä tätä rajapintaa, mikä antaa mahdollisuuden skripteistä ja HTML-sivuilta kutsuttavien C++-metodien kirjoittamiseen.

ATL HTML -kontrolli sisältää HTML-resurssin, joka toimii kontrollin esimerkki-rajapintana ja metodin, joka demonstroi HTML-käyttöliittymästä käytettävien metodien kirjoittamista.

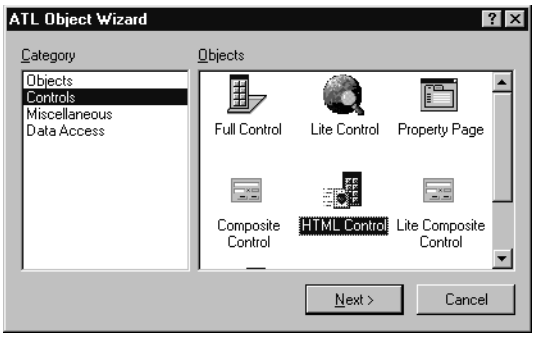
Seuraavassa harjoituksessa nähdään, kuinka tehdään yksinkertainen HTML-kontrolli käyttämällä ATL:ää.

► **MyHtmlControl-projektin luominen**

1. Valitse Visual C++:n **File**-valikosta **New**. Luo **ATL COM AppWizard** -projekti **MyHtmlControl**.
2. ATL COM AppWizardin vaiheessa 2 hyväksy oletukset ja napauta **Finish**. Luo projekti napauttamalla **OK**.

► **HTML-kontrollin lisääminen**

1. Napauta ClassViewissä hiiren kakkospainikkeella **MyHtmlControl classes** ja napauta sitten **New ATL Object**. Valitse ATL Object Wizardissa **Controls**-kategoria ja **HTML Control**, kuten seuraavan sivun kuvassa 12.8.



Kuva 12.8 ATL Object Wizard

2. Napauta **Next**. Kirjoita **ATL Object Wizard Properties** -dialogin **Names**-sivulla olevaan **Short Name** -ruutuun **MyHtmlCtrl**. Muut ruudut täyttyvät automaattisesti ja lopputuloksen tulisi olla kuvan 12.9 mukainen.

Kuva 12.9 ATL Object Wizard Properties -dialogi

3. Lisää HTML-kontrolli projektiin napauttamalla **OK**.
4. Kaksoisnapauta ResourceViewissä HTML-resurssikansiossa **IDH_MYHTMLCTRL**-resurssia, jolloin saat näkyviin esimerkki-HTML-sivun, joka on lisätty malliksi MyHtmlCtrl-kontrollin käyttöliittymästä. HTML-koodi näyttää seuraavalta:

```
<HTML>
<BODY id=theBody>
<BUTTON onclick='window.external.OnClick(theBody, "red");'>Red
</BUTTON>
<BR>
```

```
<BR>
<BUTTON onclick='window.external.OnClick(theBody, "green");'>Green
</BUTTON>
<BR>
<BR>
<BUTTON onclick='window.external.OnClick(theBody, "blue");'>Blue
</BUTTON>
</BODY>
</HTML>
```

Tämä yksinkertainen DHTML-koodi määrittelee sivulle kolme painiketta, jotka kaikki kutsuvat **onclick**-tapahtuman seurauksena **OnClick()**-funktiota. **OnClick()**-funktio saa kaksi parametriä: elementin tunnisteen ja merkkijonon, joka määrittelee yhden Internet Explorerin väritaulukon värin.

OnClick()-funktio ei kuulu DHTML-objektimalliin. Se on **IMyHtmlCtrlUI**-rajapinnan metodi, jonka määrittelyt on tehnyt ATL Object Wizard. **IMyHtmlCtrlUI**-rajapinnalle määriteltävät ominaisuudet ja metodit ovat kutsuttavissa kontrollin DHTML-koodista *window.external*-objektin kautta.

Jos tutkit MyHtmlCtrl.h-tiedostoa, näet **IMyHtmlCtrlUI::OnClick()**-toteutuksen:

```
// Example method called by the HTML to change the <BODY>
    background color
STDMETHOD(OnClick)(IDispatch* pdispBody, VARIANT varColor)
{
    CComQIPtr<IHTMLBodyElement> spBody(pdispBody);
    if (spBody != NULL)
        spBody->put_bgColor(varColor);
    return S_OK;
}
```

OnClick()-funktio saa parametreinä DHTML-sivun **IDispatch**-osoittimen ja VARIANT-arvon, joka sisältää väriarvon. Funktio hakee ATL:n **CComQIPtr**-luokkaa käyttäen osoittimen **IHTMLBodyElement**-rajapintaa. Jos **IHTMLBodyElement**-käyttöliittymäosoitin on oikea (BODY-elementti on siirretty oikein), **put_bgColor()**-metodia kutsumalla asetetaan dokumentin rungon taustaväri.

5. Käännä MyHtmlControl-projekti, jolloin luodaan MyHtmlControl.dll ja MyHtmlCtrl ActiveX-kontrolli rekisteröidään. Tarkista MyHtmlCtrl-kontrollin toiminta seuraavien harjoitusten avulla.

► **MyHtmlCtrl-kontrollin testaus**

1. Valitse Visual C++:n **Tools**-valikosta **ActiveX Control Test Container**.

2. Valitse ActiveX Control Test Containerin **Edit**-valikosta **Insert New Control**. Napauta **Insert Control** -dialogista **MyHtmlCtrl Class** ja sijoita kontrolli säilöön napauttamalla **OK**.
3. Varmista kokeilemalla, että painikkeet vaihtavat taustavärin niin kuin on ajateltu.

Huomio Tämä kontrolli ei toimi oikein kaikkien Internet Explorer 5 -versioiden kanssa. Joudut mahdollisesti asentamaan Internet Explorer 4.0:n nähdäksesi oikean toiminnan.

Seuraavissa harjoituksissa lisätään painike ja käsittelijä, joka siirtää ja avaa Web-sivun HTML-kontrolliin.

► **WWW-painikkeen lisääminen**

1. Avaa **IDH_MYHTMLCONTROL** HTML-resurssi editoriin.
2. Lisää seuraavat rivit dokumentin BODY-osan loppuun:

```
<BR>
<BR>
<BUTTON onclick='window.external.GoToWeb();'>WWW</BUTTON>
```

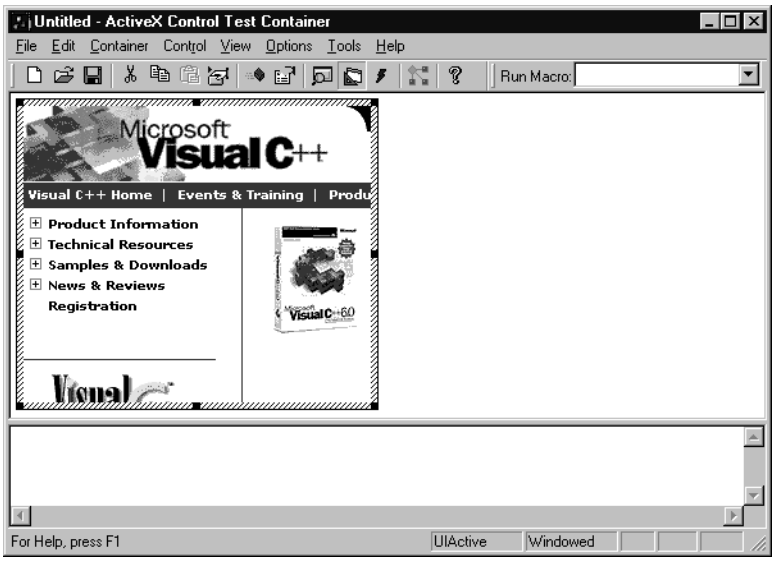
► **GoToWeb()-metodin lisääminen**

1. Avaa ClassViewissä **CMyHtmlCtrl**.
2. Napauta hiiren kakkospainikkeella **IMyHtmlCtrlUI**-rajapintayksikköä, joka on **CMyHtmlCtrl**:n alla. Valitse **Add Method**.
3. Lisää **Add Method to Interface** -dialogissa **GoToWeb**-kohtaan **Method Name**. Jätä **Parameters**-ruutu tyhjäksi ja lisää metodi napauttamalla **OK**.
4. Avaa **IMyHtmlCtrlUI**-rajapinta ja kaksoisnapauta **GoToWeb()**-funktiota, jolloin MyHtmlCtrl.cpp-tiedosto avautuu ja muokkaa **GoToWeb()**-funktion toteutusta.
5. Korvaa **GoToWeb()**-funktion //TODO-kommentti seuraavalla koodirivillä:

```
m_spBrowser->Navigate(CComBSTR("http://www.microsoft.com/visualc/"),
    NULL, NULL, NULL, NULL);
```

CMyHtmlCtrl::m_spBrowser on osoitin **IWebBrowser2**-rajapintaan, joka on WebBrowser-kontrollin Automation-rajapinta. Kontrolli saa tämän jäsenmuuttujan ATL:n Object Wizardilta.

6. Käännä MyHtmlControl-projekti ja kokeile MyHtmlCtrl-kontrollia lataamalla se ActiveX Control Test Containeriin. Tarkista, että kun napautat **WWW**-painiketta, kontrolliin avautuu Visual C++:n kotisivu kuten kuvassa 12.10.



Kuva 12.10 MyHtmlCtrl ActiveX-kontrolli

Dynamic HTML -skriptletit

DHTML skriptletit (scriptlet) esiteltiin Internet Explorer 4.0:n yhteydessä. Ne antavat mahdollisuuden yksinkertaisten käyttöliittymäkontrollien tekemiseen DHTML:llä. Skripletit ovat uudelleen käytettäviä DHTML-objekteja, joilla on hyvin määritelty julkinen rajapinta — joukko ominaisuuksia ja metodeja, jotka ovat asiakkaiden käytettävissä. Skriptletit voivat myös laukaista tapahtumia.

Skriptlettien kiintoisimpia piirteitä on se, että niitä voidaan käyttää samoin kuin ActiveX-kontrolleja. Ne voidaan esimerkiksi sijoittaa Web-sivulle tai sovelluksen lomakkeelle. Tämän mahdollistaa se, että skriptletit suoritetaan *Microsoft Scriptlet Component*in sisällä. Microsoft Scriptlet Component on ActiveX-kontrolli, joka asennetaan Internet Explorerin mukana. Sisäisesti se isännöi DHTML-parserin ilmentymää, jonka avulla skriptletin koodi tulkataan. Ulkoisesti se julkistaa ActiveX-kontrollin rajapinnan, jonka kautta voidaan olla vuorovaikutuksessa säilön kanssa.

Scriptlet Component julkistaa skriptletin käyttölliittymän kontrollisäilölle. Se toimii siltana, jonka kautta säilösovellus pääsee käsiksi skriptletin DHTML-koodissa määriteltyihin julkisiin ominaisuuksiin ja metodeihin.

Scriptlet Component sisältää ulkoisen rajapinnan, jota voidaan käyttää skriptletin koodista. Skriptlet-koodissa voidaan **window.external**-objektia käyttämällä hakea suorituksenaikaista tietoa isännän tilasta ja käyttää Scriptlet Componentin palveluja, kuten skriptletin laukaisemien tapahtumien ohjausta säilösovellukselle.

Koska skriptletien toiminta perustuu DHTML:n tulkkaamiseen, ne ovat hitaampia kuin käännetyt ActiveX-kontrollit. Kuitenkin, jos tarvitset vain kevyen kontrollin, skriptlettien avulla uudelleenkäytettävien komponenttien tekeminen on helppoa.

DHTML-skriptletin osat

DHTML-skriptletti on normaali DHTML-dokumentti, joka sisältää <BODY> ja <SCRIPT> -elementit. <BODY>-elementti määrittelee kontrollin visuaalisen ulkoasun, joka määritellään samoin kuin tavallisella Internet-sivullakin.

<SCRIPT>-elementissä määritellään ominaisuudet metodit ja tapahtumat, jotka mahdollistavat skriptlettin käyttämisen kontrollin tavoin. Tiettyjä nimeämissääntöjä noudattamalla voit erotella kontrollin käyttöliittymän kautta julkistetut ominaisuudet ja metodit skriptletin sisäisesti käyttämistä ominaisuuksista ja metodeista.

Skriptletin omaisuuksien ja metodien julkistaminen

Skriptletin kaikki ominaisuudet ja metodit, joita ei ole määritelty julkisiksi, ovat suojattuja. Kuten muutkin HTML-dokumentit, <SCRIPT>-elementit ovat paikallisia siinä dokumentissa, jossa ne on määritelty.

On kaksi tapaa, jolla julkisia ominaisuuksia ja metodeja voidaan määritellä. Yksi tapa on lisätä *public_*-etuliite kaikkien julkisten omianaisuuksien ja metodien nimien alkuun. Voit määritellä ominaisuuksia myös määrittelemällä ne funktioiksi *public_get_* ja *public_put_*-etuliitteillä. Näin voit hienosäätää ominaisuuden käsittelyä — voit esimerkiksi tehdä ominaisuudesta vain-luku-tyyppisen määrittelemällä vain **public_get()**-funktion ja jättämällä **public_put()**-funktion määrittelemättä.

Seuraavassa esimerkissä on yksinkertainen skriptletti, joka on *p1*-nimisen kappaleen sisällä. Kappaletta seuraavassa osuudessa julkistetaan kaksi ominaisuutta **text** ja **color**, joista jälkimmäinen määritellään käyttämällä **get** ja **set** -funktioita. Skriptlet julkistaa myös **animate()**-nimisen funktion.

```
<HTML><HEAD></HEAD>
<BODY>
<FONT ID="f1" color="black">
<P ID="p1">This is a paragraph of text.</P>
</FONT>

<SCRIPT LANGUAGE="JavaScript">
var public_text = p1.innerText;
function public_get_color()
{
      return f1.color;
}
function public_put_color(color)
{
      f1.color = color;
}
function public_animate()
{
      // Code to perform animation
}
</SCRIPT>
</BODY></HTML>
```

Jos asiakassivulla määritellään skriptletin ilmentymä nimellä **MyScriptlet** (tämä toimenpide käsitellään myöhemmin), seuraavat JavaScript-operaatiot tulevat mahdollisiksi:

```
MyScriptlet.text = "Hello";
…
MyScriptlet.animate();
…
var OldColor = MyScriptlet.color;
…
MyScriptlet.color ="blue";
```

Toinen keino, jolla skriptletille voidaan tehdä julkinen rajapinta, on JavaScript objektin **public_description** käyttäminen. JavaScript-muodostimessa määritellään skriptletin julkiset ominaisuudet ja metodit, sekä yhdistetään funktio skriptletin **public_description**-objektiin. Ne skriptletin ominaisuudet ja metodit, joita ei määritellä muodostimessa, eivät ole julkisia.

public_description-menetelmää käyttäen edellisen esimerkin skriptletti näyttää seuraavalta:

```
<HTML><HEAD></HEAD>
<BODY>
<FONT ID="f1" color="black">
<P ID="p1">This is a paragraph of text.</P>
</FONT>

<SCRIPT LANGUAGE="JavaScript">
public_description = new CreateScriptlet();
function CreateScriptlet()
{
this.text = p1.innerText;  // text property
this.put_color = putcolor;       // color property (write)
this.get_color = getcolor;       // color property (read)
this.animate = doAnimation;  // animate() method
}
function getcolor()
{
      return f1.color;
}
function putcolor(color)
{
      f1.color = color;
}
function doAnimation()
{
      // Code to perform animation
}
</SCRIPT>
</BODY></HTML>
```

Muodostimen nimi (joka voi olla mikä tahansa) yhdistetään **public_description**-objektiin. Äskeisessä esimerkissä, **CreateScriptlet()**-muodostin määrittelee **text** ja **color** -ominaisuudet, annetaan tekstiominaisuudelle arvo ja määritellään nimet private funktioille **getcolor()** ja **putcolor(),** joilla haetaan ja asetetaan color-ominaisuuden arvo. Funktiossa määritellään myös **animate()-**metodi ja nimetään private funktio **doAnimation()**, jossa metodi toteutetaan.

public_description-objektin käyttö on kokeneiden olio-ohjelmoijien suosima menetelmä. Tällä tekniikalla saadaan selvästi eroteltua käyttöliittymä ja sen toeteutus. Lisäksi se mahdollistaa käyttöliittymän määrittelyn siististi yhdessä paikassa — yleensä <SCRIPT>-lohkon alussa.

Skriptletien tapahtumat

Sovellus, joka sisältää skriptletin voi, vastaanottaa tietoja skriptletin tapahtumista. DHTML-skriptletti voi julkistaa seuraavanlaisiatapahtumia:

- **Standardeihin DHTML-taphtumiin** kuuluvat **onclick**-tapahtuma ja **onkeypress**-tapahtuma.
- **Mukautetut tapahtumat** (Custom events) ovat ohjelmoijan määrittelemiä tapahtumia tai standarditapahtumiin kuulumattomia DHTML-tapahtumia. Skriptletti voi laukaista tapahtuman esimerkiksi silloin, kun ominaisuuden arvo muuttuu.

► Standarditapahtumien käsittely

DHTML-skriptletti voi julkistaa minkä tahansa näistä standardi-DHTML-tapahtumista:

- onclick
- ondblclick
- onkeypress
- onkeydown
- onkeyup
- onmousemove
- onmousedown
- onmouseup

Johdannossa DHTML-skriptletteihin todettiin, että yksi Scriptlet Componentin tehtävistä on ohjata skriptletin laukaisemat tapahtumat kontrollisäilölle. Oletetaan esimerkiksi, että kaksoisnapautat skriptletin runkoa. Scriptlet Component kääntää DHTML **ondblclick**-tapahtuman ja se voi reagoida välittämällä tapahtuman säilön vastaavaksi tapahtumaksi. On kuitenkin tärkeää muistaa, että näin menetellään *vain, jos* skriptletissä on nimenomaisesti näin määrätty.

Tapahtuman välittäminen skriptletistä säilösovellukselle edellyttää tapahtuma-käsittelijän määrittelemistä siirrettävälle tapahtumalle ja tapahtumakäsittelijän **bubbleEvent**-metodin kutsumista. **bubbleEvent**-metodin tarjoaa Scriptlet Component, ja sitä voidaan käyttää **window.external**-objektin kautta.

Jatkaaksemme esimerkkiä, hiiren kaksoisnapautus käsitellään skriptletin rungossa määrittelemällä runko seuraavasti:

```
<BODY ondblclick = passthru()>
...
</BODY>
```

Tapahtuma välitetään säilölle määrittelemällä **passthru()**-funktio seuraavasti:

```
function passthru()
{
    if(!window.external.frozen)
        window.external.bubbleEvent();
}
```

Huomaa, kuinka rutiini tarkistaa Scriptlet Componentin *frozen*-ominaisuuden avulla, onko se valmiina käsittelemään tapahtumia ennen kuin välittää tapahtuman eteen päin.

► **Mukautettujen tapahtumien lisääminen ja käsitteleminen**

Standarditapahtumien käsittelemisen lisäksi voidaan skriptletistä laukaista mukautettuja tapahtumia. Mukautettuja tapahtumia voidaan käyttää:

- Lisätietojen välittämiseen standarditapahtumista — esimerkiksi mitä skriptletin painikkeista on napautettu.
- Ilmoitettaessa isäntäsovellukselle skriptletin epästandardeista muutoksista, esimerkiksi ominaisuuden arvon muuttuessa.
- Ilmoitettaessa isäntäsovellukselle DHTML-tapahtumista, jotka eivät kuulu **bubbleEvent**-metodin käsittelemiin standarditapahtumiin.

Voit laukaista mukautettuja tapahtumia mistä kohdasta skriptlettiä vain käyttämällä **raiseEvent**-metodia. Seuraava esimerkki näyttää, kuinka mukautettua tapahtumaa käyttäen ilmoitetaan säilösovellukselle skriptletin taustavärin muuttumisesta:

```
<SCRIPT LANGUAGE="JavaScript">
function public_put_backgroundColor(value)
{
    window.document.bgColor = value;
    if(!window.external.frozen)

    window.external.raiseEvent("onbgcolorchange",
        value);
}
</SCRIPT>
```

Huomaat, että **raiseEvent**-metodi saa kaksi parametriä. Ensimmäinen parametri määrittelee tapahtuman nimen, jolloin sille saadaan standardi rajapinta säilöä varten. Scriptlet Component reitittää kaikki mukautetut tapahtumat yhden **onscriptletevent**-nimisen tapahtuman kautta. Laukaistu tapahtuma voidaan selvittää *name*-parametrin avulla. Toista parametriä käytetään tapahtumaan liittyvän tiedon välittämiseen.

Seuraavassa esimerkissä nähdään, kuinka Internet Explorer käsittelee edellisessä esimerkissä laukaistun **onbgcolorchange**-tapahtuman:

```
<SCRIPT LANGUAGE=”JavaScript”
      FOR=”MyScriptlet”
      EVENT=”onscriptletevent (event, obj)”>
      if(event == “onbgcolorchange”)
      {
            msg = “Scriptlet background changed to “ + obj;
            alert(msg);
      }
</SCRIPT>
```

Skriptlet-esimerkki

Jotta ymmärtäisit, kuinka skriptletit toimivat ja kuinka voit niitä sovelluksissasi käyttää, otamme malliksi **ColorSelector**-skriptletin, joka on yksinkertainen skriptletti, jonka avulla voit selata värejä ja valita niistä yhden. Tämän skriptletin lähdekoodi on oheisrompulla tiedostossa Chapter 2\Exercises\ColorSelector.htm:

```
<!— ColorSelector.htm —>
<HTML><HEAD></HEAD>
<BODY style=”border:2px solid black;
      color:white; background-color:white “
      ondblclick = passthru()>

<DIV ID=d1 style=”position:relative; border:1px solid black;
top:5; left:5; width:130; height:50">
</DIV>

<P ID=p1 style=”position:relative; left:13">
<BUTTON ID=BackButton onclick = cycle(“back”)> < </BUTTON>.
<BUTTON ID=ForwardButton onclick = cycle(“forward”)> > </BUTTON>
<BUTTON ID=SelectButton onclick = doSelect()> Select </BUTTON>
</P>
</BODY>

<SCRIPT LANGUAGE=”JavaScript”>
```

```
public_description = new colorselector();
var colors = new Array("white", "red", "green", "blue", "black") ;
var numcolors = colors.length;
var curcolor = 0;

// Object description
function colorselector()
{
    this.get_color = getcolor;
}

// Public property access function
function getcolor()
{
    return colors[curcolor];
}

// Events
// Standard
function passthru()
{
    if(!window.external.frozen)
        window.external.bubbleEvent();
}

// Custom
function doSelect()
{
    if(!window.external.frozen)
        window.external.raiseEvent("onClickSelect", colors[curcolor]);
}

// Private methods
function cycle(direction)
{
    if(direction == "back")
    {
        curcolor — ;
        if(curcolor < 0)
            curcolor = numcolors-1;
    }
    if(direction == "forward")
    {
        curcolor++;
        if(curcolor >= numcolors)
            curcolor = 0;
    }
```

```
    d1.style.backgroundColor = colors[curcolor];
}
</SCRIPT>
</HTML>
```

Tutkimalla skriptletin **public_ description**-objektiin liitettyä **colorselector()**-funktiota näet, että skriptletti julkistaa yhden vain-lukuominaisuuden, jonka nimi on **color**. Huomaa, että skriptlet julkistaa standarditapahtuman **ondblclick** ja laukaisee mukautetun tapahtuman **onClickSelect**.

Skriptlettejä isännöivät Web-sivut

Skriptletti voidaan sijoittaa Web-sivulle tavallisen ActiveX-kontrollin tapaan käyttämällä HTML:n <OBJECT>-elementtiä. Tavallisista ActiveX-kontrolleista poiketen, skriplettejä ei yksilöidä ClassID:n avulla. Sen sijaan skriptletin **MIME**-tyyppi yhdistetään **TYPE**-attribuuttiin, ja skriptletin lähdekooditiedostoon viittaava URL sijoitetaan **DATA**-attribuuttiin. MIME (Multipurpose Internet Mail Extensions) on Internetprotokolla, jonka avulla selain voi tunnistaa esittämänsä tiedoston tyypin. **b**-tyyppi, joka määrittelee dokumentin skriptletiksi on *text/x-scriptlet*.

Seuraavasta esimerkistä nähdään, kuinka tiedostossa MyScriptlet.htm määritelty skriptletti otetaan käyttöön.

```
<OBJECT ID="MyScriptlet" TYPE="text/x-scriptlet"DATA="MyScriptlet.htm">
</OBJECT>
```

Tässä koodissa oletetaan, että MyScriptlet.htm-tiedosto sijaitsee samassa hakemistossa avoimen tiedoston kanssa.

Olemme tehneet **ColorSelector**-skriptletin kokeilemista varten yksinkertaisen HTML-sivun. *TestScriptlet.htm* on kansiossa Chapter 12\Exercises ja sen listaus näyttää seuraavalta:

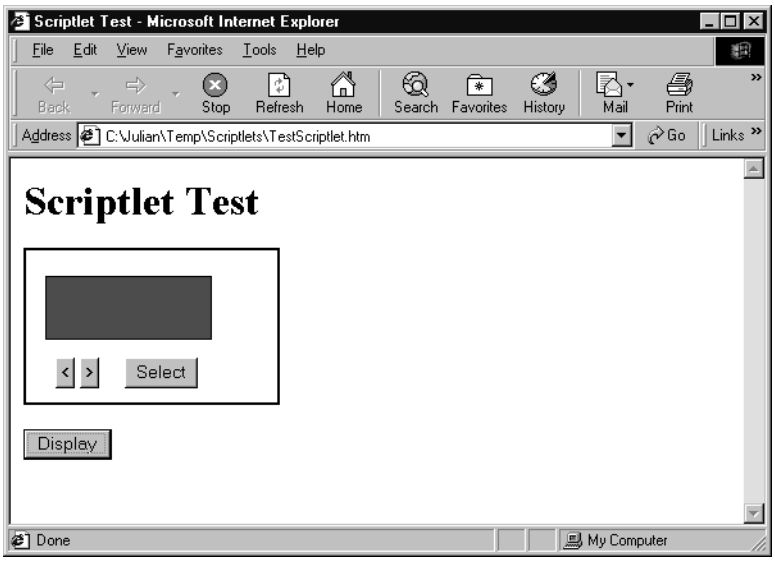
```
<!— TestScriptlet.htm —>
<HTML>
<HEAD>
<TITLE>
Scriptlet Test
</TITLE>
</HEAD>
<BODY>
<H1> Scriptlet Test
</H1>
<P>
<OBJECT ID=”MyScriptlet” TYPE=”text/x-scriptlet”
    DATA=”ColorSelector.htm”>

</OBJECT>
```

```
</P>
<P>
<BUTTON ID="Btn1" onclick="UseScriptlet()"> Display </BUTTON>
</P>
</BODY>

<SCRIPT LANGUAGE="JavaScript">
function UseScriptlet()
{
    alert(MyScriptlet.color);
}
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript"
    FOR="MyScriptlet"
    EVENT="onscriptletevent(event, obj)">
    if(event == "onClickSelect")
        document.body.bgColor = obj;
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript"
    FOR="MyScriptlet"
    EVENT=ondblclick>
    alert("double-clicked in scriptlet");
</SCRIPT>
</HTML>
```

Tällä sivulla voidaan testata skriptletin toiminnallisuutta seuraavin tavoin:

- Skriptlettiä isännöi Internet Explorer. Voit kokeilla skriptletin julkisia funktioita selaamalla sen värejä nuolipainikkeilla.
- Kun napauta testisivun **Display**-painiketta, skriptletti hakee **color**-ominaisuutensa nykyisen arvon ja näyttää sen viestiruudussa.
- Kun napautat skriptletin **Select**-painiketta, skriptletti laukaisee **onClickSelect**-tapahtuman. Tapahtuman käsittelijä asettaa parametrinä välitetyn värin testisivun taustaväriksi.
- Testisivulla on käsittelijä, joka huolehtii skriptletin standarditapahtuman **ondblclick**.

Avaa TestScriptlet.htm-tiedosto Internet Exploreriin ja kokeile skriptlettiä. Kuvassa 12.11 seuraavalla sivulla näet skriptletin toiminnassa.



Kuva 12.11 **ColorSelector**-skriptletin testaus

Skriptlettien sijoittaminen Visual C++ -sovellukseen

Vaikka skriptletti voidaan sijoittaa Microsoft Visual Basic -sovelluksen lomakkeelle ja sitä voidaan käyttää tavallisen ActiveX-kontrollin tapaan, tämä toiminto ei ole vielä mahdollinen Visual C++ säilöille. Teoriassa Scriptlet Component komponentti voidaan lisätä säilöön käyttämällä Components and Controls gallerya, jolloin luodaan **CWebBridge**-luokka, joka käärii komponentin julkistaman **IWebBridge**-rajapinnan. Voit asettaa tämän luokan avulla Scriptlet komponentin **url**-ominaisuudelle arvon, joka sisältää skriptletin HTML-dokumentin URL:n ja voit käsitellä skriptletin mukautettuja tapahtumia skriptletin **event**-ominaisuuden kautta.

Vaikka Scriptlet Component -komponentin sijoittaminen projektiin onnistuu ja **CWebBridge**-luokka saadaan muodostettua, kontrollia ei voida tällä hetkellä sijoittaa dialogiin. Jos yrität, seurauksena on kuvassa 12.12 näkyvä ilmoitus.

Kuva 12.12 Scriptlet Component -komponentin virheilmoitus

Käyttäjät, jotka yrittävät käyttää **CWebBridge**-luokkaa Scriptlet Component -kontrollin ilmentymän luomiseen, kohtaavat Knowledge Basen artikkelissa Q190838 kuvatun kaltaisia ongelmia. Artikkelissa mainitaan, että voit ehkä luoda Scriptlet Component -komponentin ilmentymän tuomalla (#import-komennon avulla) tyypikirjaston. Siihen saakka, kunnes toiminto, jolla Scriptlet Component voidaan sijoittaa dialogiin, tulee saapuville, on parasta sijoittaa skriptletti HTML-sivulle ja sijoittaa sivu **CHtmlView**-pohjaiseen sovellukseen tai ATL:n HTML-kontrolliin.

Oppitunnin yhteenveto

Internetin nopea kasvu on muuttanut monella tapaa näkemyksiä sovellusten kehittämisestä. Web-selaimet kuten Internet Explorer eivät ole vain ohjelmia, joita käytettään World Wide Web -sivujen tai yhtiön intranetsivujen tutkimiseen, vaan niillä voidaan käsitellä lähiverkon palvelimilla ja omalla tietokoneella olevia resursseja ja ajaa sovelluksia. Voit tehdä selaintyylisiä sovelluksia, jotka HTML:n avulla tuottavat monipuolisen ja yhdenmukaisen käyttöliittymän tarjoten saumattoman pääsyn omalla tietokoneella, yrityksen palvelimilla ja ympäri maailmaa sijaitsevilla Web-palvelimilla oleviin resursseihin.

HTML järjestelmä, jossa ASCII-dokumenttiin lisätään muotoilukomentoja viittauksia grafiikka-, ääni- ja animaatiotiedostoihin sekä hyperlinkkejä Web-sivuille ja muihin resursseihin. HTML:ssä määritellään joukko standardeja muotoilukomentoja, jotka kaikki selaimet tunnistavat. Nämä merkinnät jakavat HTML-dokumentin erillisiin osiin, joita kutsutaan elementeiksi. Osa merkinnöistä vaikuttaa tekstin muotoiluun, toisia käytetään, kun dokumenttiin halutaan lisätä ulkoisia resursseja kuten kuvatiedostoja, tai kun määritellään hyperlinkkejä toisiin dokumentteihin. Useimpiin HTML-merkintöihin voidaan liittää attribuutteja, joiden avulla elementistä voidaan antaa lisätietoja.

DHTML:n avulla voidaan luoda dynaamisia ja vuorovaikutteisia Web-sivuja. DHTML:ää käyttämällä voidaan esimerkiksi muuttaa Web-sivun sisältöä ja ulkoasua, sekä käsitellä käyttäjän syöttämiä tietoja. DHTML tekee nämä muutokset käyttäjän tietokoneella muodostamalla ja näyttämällä muuttuneen sivun uudelleen.

DHTML-dokumentti käyttää HTML-tyylejä määritellessään dokumentin muotoa ja ulkoasua, sekä lyhyitä skriptilohkoja, joiden avulla se käsittelee käyttäjän syöttämät tiedot ja muuttaa sivun muotoiluja ja tekstejä käsitelemällä suoraan HTML-muotoiluja. DHTML-dokumentin skriptit hyödyntävät DHTML:n objektimallia käsitellessään näytettävän sivun elementtejä.

Microsoft tarjoaa käyttettäväksi WebBrowser ActiveX-kontrollin, joka auttaa Internet Explorer Web-selaimen toimintojen lisäämisessä sovellukseen. MFC:n luokka **CHtmlView** sisällyttää WebBrowser-kontrollin MFC:n dokumentti/näkymä-arkkitehtuuriin. **HtmlView**-luokka sisältää **GetHtmlDocument()**-

funktion, palauttaa näkyvillä olevan HTML-dokumentin erillisen rajapinnan osoittimen. Tämän osoittimen avulla voidaan käyttää DHTML-objektimallin Automation-rajapintoja ja käsitellä näin dokumenttia.

ATL:ää käyttämällä voidaan luoda kontrolli, joka pystyy näyttämään HTML-sivuja. ATL:n HTML-kontrolli määritelee erillisen rajapinnan, johon voidaan lisätä metodeja ja ominaisuuksia, jotka määrittelevät, kuinka kontrolli on yhteydessä säilöön. ATL:n HTML-kontrolli määrittelee myös rajapinnan, jota käytetään C++-koodin ja HTML-käyttöliittymän välillä. Kontrollissa näytettävässä HTML-sivussa oleva skripti voi käyttää näitä ominaisuuksia ja metodeja DHTML:n **window.external**-objektin kautta.

Voit lisätä HTML-sivun resurssina projektiisi, samoin kuin bittikartan tai dialogimallin. HTML-resurssit käännetään sovelluksen suoritettavaan tiedostoon, joten näin voidaan olla varmoja, että sovelluksen käyttämät HTML-sivut ovat aina saatavilla. Voit käyttää URL:ssä res:-protokollaa ilmoittaessasi suorittetavassa tiedostossa tai DLL:ssä sijaitsevan resurssin sijaintia.

DHTML-skriptletit ovat uudelleenkäytettäviä DHTML-objekteja, jotka julkistavat ominaisuuksia ja metodeja, joita asiakkaat voivat käyttää. Skriptlettejä voidaan käyttää kuten ActiveX-kontrolleja — ne voidaan esimerkiksi sijoittaa Web-sivulle tai C++-sovelluksen dialogiin. Skriptletit suoritetaan Scriptlet Component -komponentissa, joka on Internet Explorerin mukana asennettu ActiveX-kontrolli. Scriptlet Component toimii siltana niin, että säilösovellus voi käsitellä skriptletin DHTML-koodissa määriteltyjä julkisia ominaisuuksia ja metodeja.

Scriptlet Component sisältää ulkoisen rajapinnan, jota voidaan käsitellä skriptletin koodista. Skriptletin koodissa voidaan tarkkailla säilön tilaa tekemällä kyselyjä **window.external-**objektin avulla. Sen avulla voidaan käyttää myös Scriptlet Component -komponentin palveluja, kuten esimerkiksi skriptletin sisältävän sovelluksen tapahtumien edelleenohjausta.

Oppitunti 2: ActiveX-dokumenttien luominen

ActiveX-dokumentit (myös *Active documents),* ovat *ActiveX-dokumenttipalvelinsovellusten* luomia dokumentteja. ActiveX-dokumentteja voi katsella ja muokata myös muilla kuin ne luoneilla sovelluksilla. Tarkoitukseen käyvät kaikki *ActiveX-dokumenttisäilösovellukset*.

Microsoft Word -dokumentit ovat esimerkiksi ActiveX-dokumentteja. Internet Explorerin (ActiveX-dokumenttisäilö) käyttäjät voivat katsella ja muokata Word-dokumentteja aivan samoin kuin Wordissäkin (ActiveX-dokumenttipalvelinsovellus) Internet Explorerin selainikkunasta.

ActiveX-dokumenteista on tullut tehokas ominaisuus Microsoft-sovellusten käyttäjien Web-sivuille. Ne tarjoavat laajennuksia Internet Explorer -selaimen toimintoihin ja monipuolisen vaihtoehdon yksinkertaisille HTML-sivuille.

Tämän oppitunnin jälkeen:

- Tiedät, mitä ovat ActiveX-dokumentit.
- Tiedät, mitä etuja ActiveX-dokumenttien käyttämisestä Web-sivuilla saadaan.
- Tiedät, kuinka MFC AppWizardia käytetään ActiveX-dokumenttipalvelimen luomiseen.
- Tiedät, kuinka ActiveX-dokumentteja jaellaan Web-sivuilla, ja kuinka ActiveX-dokumentteja katsellaan Internet Explorerilla.

Oppitunnin arvioitu kesto: 30 minuuttia

ActiveX-dokumenttien käyttäminen

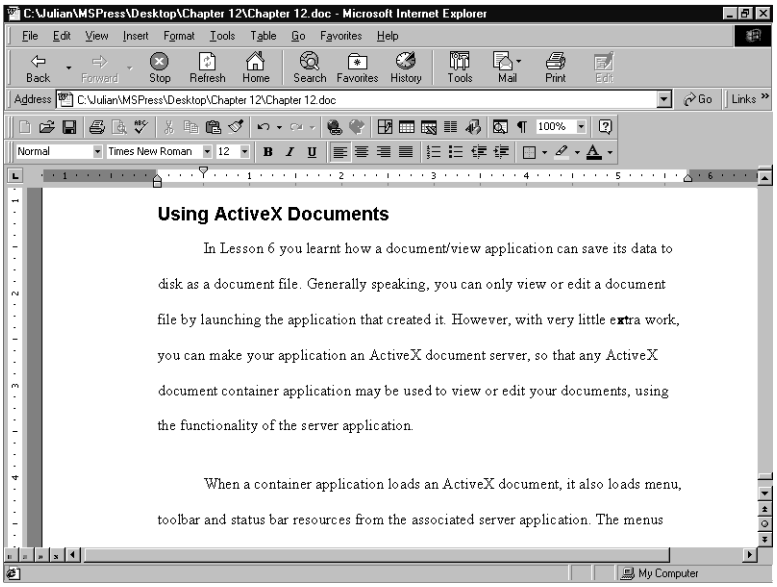
Luvussa 6 opit, kuinka dokumentti/näkymä-sovellus voi tallentaa tietonsa levylle dokumenttitiedostoon. Yleensä voit katsella ja muokata dokumenttia vain käynnistämällä sovelluksen, jolla se on luotu. Kuitenkin voit hyvin pienenllä vaivalla tehdä sovelluksesta ActiveX-dokumenttipalvelimen, jolloin mitä tahansa ActiveX-dokumenttisäilöä voidaan käyttää dokumentin katselemiseen ja muokkaamiseen siinä kuin palvelinsovellustakin.

Kun säilösovellus lataa ActiveX-dokumentin, se lataa myös palvelinsovellukseen liittyvät valikko-, työkalurivi- ja tilariviresurssit. Valikot ja työkalurivit yhdistetään säilön kehysikkunan vastaaviin, jolloin palvelinsovelluksen toiminnot tulevat käyttöön myös säilösovelluksessa. Tämän seurauksena palvelin “valtaa” hetkeksi säilön ActiveX-dokumenttia näytettäessä.

Jos Internet Explorer ja Word ovat asennettuina, voit helposti kokeilla ActiveX-dokumenttitekniikkaa.

► **Kiintolevyllä olevien ActiveX-dokumenttien katseleminen**

1. Avaa Internet Explorer.
2. Avaa Windowsin Resurssienhallinta ja etsi Wordin .doc-tiedosto.
3. Raahaa Word-dokumenttitiedosto Resurssienhallinnasta Internet Explorerin ikkunaan. Word-dokumentin tulisi avautua Internet Exploreriin ActiveX-dokumenttina, kuten kuvassa 12.13.



Kuva 12.13 ActiveX-dokumentin katseleminen Internet Explorer 5:llä

Huomaa, että Internet Explorer näyttää Wordin valikot omien valikkojensa lisäksi. Kokeile Word-dokumentin muokkaamista Internet Explorerissa, ja kokeile muutamia Word-valikoiden toiminnoista. Palaa edelliselle sivulle napauttamalla **Back**-painiketta. Internet Explorer kysyy, haluatko tallentaa muokatun ActiveX-dokumentin ennen takaisin siirtymistä.

Huomio Internet Explorerin versioiden 4.0 ja 5.0 tavassa käsitellä ActiveX-dokumentteja on joitakin eroja. Internet Explorer 4.0:ssä voit avata ActiveX-dokumentin määrittelemällä sen *file://* URL:n osoiterivillä. Internet Explorer 5 puolestaan avaa yleensä Wordin dokumentin katselemista varten.

ActiveX-dokumenttien käyttäminen Web-palvelussa

Internet Explorer ei ole ainoa ActiveX-dokumenttisäilösovellus (Microsoft Office Nitoja on toinen), mutta se on säiliöistä se, joka sopii tämän luvun teemaan, koska käsittelemme Internet-ohjelmointia. Äskeisessä esimerkissä

kuvattu tilanne oli jossain määrin keinotekoinen. Internet Exploreria ei yleensä käytetä ActiveX-dokumenttien katselemiseen omalla tietokoneella — on paljon tehokkaampaa käyttää dokumentin omaa isäntäsovellusta.

Internet Explorer käyttää ActiveX-säilötoimintojaan Web-palvelimelta jaettujen ActiveX-dokumenttien esittämiseen. Tämä tarkoittaa, että Wordin asiakirjat, Microsoft Excelin taulukot ja mitkä tahansa muut ActiveX-dokumentit, jotka on sisällytetty Web-palveluun, ovat kenen tahansa katseltavissa ja muokattavissa, jos käyttäjällä on käytettävissä vastaava ActiveX-palvelinsovellus omalla tietokoneellaan. ActiveX-palvelinsovellus ei salli tietojen tallentamista takaisin palvelinkoneelle, mutta antaa tallentaa tiedot koneen omalle kiintolevylle. Toisin sanoen **File**-valikon **Save**-komento ei ole käytettävissä, mutta **Save As** -komento on.

ActiveX-dokumetteja käytetään usein, kun halutaan tehdä “lomakkeita” yrityksen intranetpalveluun. Intranetissä voi olla kokoelma työntekijöiden usein käyttämiä lomakkeita — kuten koulutusanomuksia tai matkakorvauslomakkeita — Word tai Excel -muodossa. Näin työntekijät voivat ladata tyhjän lomakkeen selaimelleen, olivatpa he sitten toimistossa, kotonaan tai toisella puolella maapalloa. ActiveX-dokumenttitekniikkaa käyttäen he voivat täyttää lomakkeen ja tallentaa sen sitten paikallisena kopiona tai lähettää faksina tai sähköpostina henkilöstöosastolle.

Web-sivustosta voidaan tehdä ActiveX-dokumenttitekniikan avulla huomattavasti monipuolisempi kuin perus-HTML:llä. Käytettäessä Excel-taulukoita Webin kautta voidaan tietoja järjestellä ja esittää kaikkien Excelin tehokkaiden toimintojen avulla. Rajoituksena voidaan pitää sitä, että ActiveX-palvelinsovelluksen täytyy olla asennettuna paikalliselle tietokoneelle ennen kuin Web-sivulla olevia ActiveX-dokumentteja voidaan tutkia. Tätä ongelmaa voidaan yrittää ratkaista tekemällä palvelinsovelluksesta eri versioita. Voidaan esimerkiksi tehdä täysversio, jota käytetään Web-sivuilla olevien ActiveX-dokumenttien luomiseen ja muokkaamiseen ja kevennetty versio, jota jaetaan Internet-asiakkaille käytettäväksi Web-selaimen kanssa. Microsoft käyttää tätä lähestymistapaa tarjotessaan Word-dokumenttien tutkimiseen soveltuvan katseluohjelman, joka mahdollistaa Web-sivuilla olevien Word-dokumettien lukemisen vaikka Wordiä ei koneelle olisi asennettukaan.

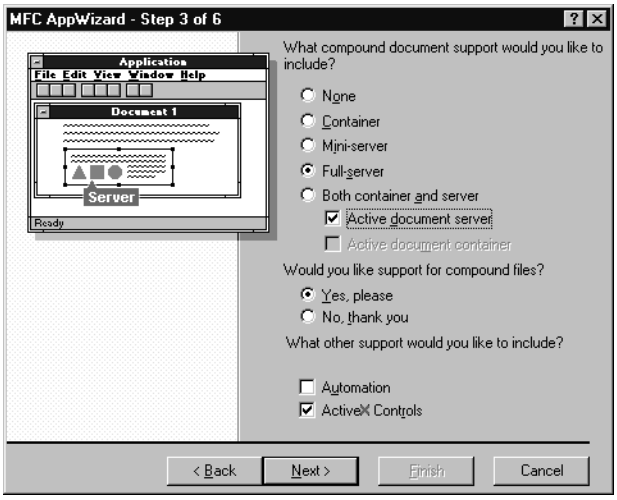
Yksi mahdollinen ongelma ActiveX-dokumenttitekniikan käyttämisestä Web-sivuilla on se, että käyttäjät ovat sidottuja selaimiin, jotka tukevat ActiveX-dokumetteja. Käyttäjillä täytyy olla myös kopio ActiveX-palvelinsovelluksesta koneellaan, joten heidän täytyy näin ollen käyttää Windowsia. Tästä johtuen ActiveX-dokumenttien käyttäminen soveltuu parhaiten tilanteisiin, joissa käyttäjien tietokoneiden käyttöjärjestelmäympäristö ja asetukset ovat määrättävissä. Kun halutaan tehdä julkinen Web-sivusto, jolla on mahdollisimman paljon ennalta tuntemattomia käyttäjiä, on parasta käyttää HTML:ää.

ActiveX-dokumenttipalvelimen luominen

Visual C++ 6.0:lla ActiveX-dokumenttien tekeminen on todella helppoa. Kun AppWizardissa valitaan sopiva toiminto, se luo kaikki ActiveX-dokumenttipalvelimen tekemiseen tarvittavat koodit ja resurssit.

► **ActiveX-dokumenttipalvelimen luominen**

1. Luo AppWizardilla uusi .exe-projekti **MyADSApp**. Valitse vaiheessa 1 **Single document** -vaihtoehto.
2. Hyväksy oletukset vaiheessa 2. Valitse vaiheessa 3 **Full-server** ja **Active document server** -vaihtoehdot, kuten kuvassa 12.14.



Kuva 12.14 MFC AppWizard -yhdistelmäasiakirjavalinnat

3. Napauta vaiheessa 4 **Advanced** ja kirjoita tarkentimeksi **ads**. Sulje **Advanced Options** -dialogi.
4. Napauta **Finish** ja luo sitten projekti napauttamalla **OK**.

Seuraavassa harjoituksessa tehdään yksinkertainen sovellus, joka näyttää sovelluksen pääikkunassa tekstirivin. Ensin tehdään dialogi, jonka avulla käyttäjä voi määrittää näytettävän tekstin.

► **Set Display String -dialogin tekeminen**

1. Luo dialogimalli, jossa on yksi muokkausruutukontrolli, kuten kuvassa 12.15. Dialogin tunniste on **IDD_EDITSTRING** ja muokkausruudun **IDC_NEWSTRING**.

Kuva 12.15 Set Display String -dialogi

2. Luo **CEditStringDlg** niminen dialogi. Luo ClassWizardin **Member Variables** -sivulla **Value**-kategorian **CString**-jäsenmuuttuja m_newstring.

Seuraavaksi luodaan sovelluksen tiedot (rivi, joka näytetään) lisäämällä dokumenttiluokkaan tietojäsen. Tietojen tallentamista ja hakemista varten tehdään myös **Serialize()**-funktio.

► **CMyADSApp-sovelluksen tietojen luominen**

Lisää julkinen **CString**-jäsenmuuttuja m_strDisplay **CMyADSAppDoc**-luokkaan. Muokkaa **CMyADSAppDoc::Serialize()**-funktiota niin, että se serialisoi m_strDisplay-muuttujan seuraavassa koodissa nähtävällä tavalla:

```
void CMyADSAppDoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        ar << m_strDisplay;
    }
    else
    {
        ar >> m_strDisplay;
    }
}
```

Seuraavaksi lisätään sovelluksen valikkoon ja työkaluriville **Edit String** -komento. Komennolle tehdään myös käsittelijä.

► **Edit String -valikkokomennon luominen ja toteutus**

1. Muokkaa **IDR_MAINFRAME**-valikkoresurssia. Poista **Edit**-valikosta **Undo** -komento. Korvaa se uudella komennolla **&String**. Lisää sopiva kehoterivi. Hyväksy komennon oletustunniste **ID_EDIT_STRING**.
2. Muokkaa **IDR_MAINFRAME**-työkaluriviä. Luo tekstin lisäämistä edustava työkalupainike. Liitä uuteen painikkeeseen **ID_EDIT_STRING**-tunniste.
3. Lisää seuraava rivi MyADSAppDoc.cpp-tiedoston alkuun:

```
#include “EditStringDlg.h”
```

4. Luo **CMyADSAppDoc**-luokkaan käsittelijä **ID_EDIT_ STRING**-tunnisteelle. Anna funktion nimeksi **OnEditString()** ja toteuta funktio seuraavasti:

```
void CMyADSAppDoc::OnEditString()
{
    CEditStringDlg myDialog;
    myDialog.m_newstring = m_strDisplay;

    if(myDialog.DoModal() == IDOK &&
        myDialog.m_newstring != m_strDisplay)
    {
        m_strDisplay = myDialog.m_newstring;
        UpdateAllViews(NULL);
        SetModifiedFlag();
    }
}
```

Tämä funktio avaa **Set Display String** -dialogin, jonka avulla käyttäjä voi määritellä sovelluksen näyttämän rivin. Rivi talletetaan CMyADSAppDoc:: m_strDisplay-muuttujaan.

► **Sovelluksen piirtofunktion luominen**

Etsi **CMyADSAppView::OnDraw()**-funktio. Korvaa //TODO-kommentti seuraavalla koodilla:

```
CFont HeadingFont;
if(HeadingFont. CreateFont(64, 0, 0,
    0, FW_NORMAL, 1, 0, 0, 0, 0, 0, FF_DECORATIVE, 0))

    pDC->SelectObject(&HeadingFont);

pDC->TextOut(10, 10, pDoc->m_strDisplay);
```

Jos käännät ja käynnistät CMyADSApp-sovelluksen tässä vaiheessa, se toimii täysin itsenäisenä sovelluksena. Voit valita **String**-komennon **Edit**-valikosta ja voit lisätä pääikkunassa lisättävän tekstirivin. Voit tallentaa rivin .ads-tarkentimelliseen dokumenttitiedostoon.

Jos kokeilet tallennetun .ads-tiedoston avaamista ActiveX-dokumenttisäilöön kuten Internet Exploreriin, näet kyllä tekstirivin, mutta et voi käyttää **Edit**-valikon **String**-komentoa. Tämä johtuu siitä, että ActiveX-dokumenttisäilö ei lataa valikkokomentoja ja työkalupainikkeita **IDR_MAINFRAME**-resurssista.

Kun ActiveX-dokumenttipalvelimen luomiseen käytetään AppWizardia, luodaan projektiin ylimääräiset valikko- ja työkaluriviresurssit, joiden molempien tunniste on ID **IDR_SRVR_INPLACE**. **IDR_SRVR_INPLACE**-valikossa olevat komennot liitetään säilön valikoihin ja **IDR_SRVR_INPLACE**-työkalurivi lisätään säilösovelluksen näyttämiin työkaluriveihin.

Jotta **Edit**-valikon **String**-komento olisi käytettävissä ActiveX-säilösovelluksessa, täytyy **IDR_SRVR_INPLACE**-valikkoa ja työkaluriviä muokata.

► **IDR_SRVR_INPLACE-valikko ja työkaluriviresurssien muokkaaminen**

1. Kaksoisnapauta ResourceViewissä **IDR_SRVR_INPLACE**-valikkoa. Muokkaa sen **Edit**-valikon **Undo**-komennon ominaisuuksia. Muuta otsikoksi **&String** ja tunnisteeksi **ID_EDIT_STRING**. Kehoterivi muuttuu automaattisesti, kun **Menu Item Properties** -dialogi suljetaan.
2. Muokkaa **IDR_SRVR_INPLACE**-työkaluriviresurssia. Luo uusi painike, joka on samanlainen kuin **IDR_MAINFRAME**-resurssille luomasi painike (voit kopioida ja liittää sen ikkunasta toiseen). Yhdistä painike tunnisteeseen **ID_EDIT_STRING**.
3. Käännä ja käynnistä CMyADSApp-sovellus. Käytä **Edit**-valikon **String**-komentoa niin, että sovelluksen pääikkunassa näkyy teksti “Hello World!” Sulje sovellus ja tallenna kehotettaessa tiedosto nimellä **Hello.ads**.
4. Käynnistä Internet Explorer. Etsi Resurssienhallintaohjelmalla Hello.ads-tiedosto ja vedä se Internet Explorerin ikkunaan. Jos **File Download** -sanomaikkuna avautuu, valitse **Open this file from the current location** ja napauta **OK**. Sivun tulisi avautua kuten kuvassa 12.16.

Kuva 12.16 Hello.ads-dokumentti Internet Explorer 5:ssa

5. Varmista, että voit **Edit**-valikon **String**-komentoa käyttämällä muuttaa dokumentin näyttämän tekstirivin. Palaa selaimessa edelliselle sivulle napauttamalla **Back**-painiketta. Valitse **No** sanomaikkunassa, jossa kysytään, haluatko tallentaa muutokset.

ActiveX-dokumentin jakelu Web-sivuilla

Seuraavassa harjoituksessa nähdään, kuinka Hello.ads-tiedostoa voidaan levittää Web-palvelussa niin, että sitä voidaan käsitellä HTTP:tä käyttämällä Internet-yhteyden yli. Ohjeet ovat IIS 4.0:lle ja PWS:lle.

Web-palvelun rakenne

Asennettaessa ensimmäistä kertaa IIS 4.0 ja PWS luovat tietokoneelle oletus-Web-palvelun, jota voidaan käyttää määrittämällä URL:ksi tietokoneen nimi. Jos kirjoitat esimerkiksi seuraavan URL:n Internet Explorerin **Address**-ruutuun, selain näyttää Web-palvelimen oletus etusivun:

```
http://[your computer name]/
```

Tietokoneen nimen perässä oleva jakoviiva (jonka selain lisää automaattisesti, jos et tee sitä itse) tarkoittaa, että halutaan mennä kotihakemistoon. Windows NT Option Packin tervetuloa sivu avautuu, koska se on määritelty kotihakemis-ton oletusdokumentiksi.

Web-sivusto voidaan organisoida *virtuaalihakemistojen* (virtual directory) hierarkiaksi. Virtuaalihakemistot ovat Web-sivuston rakenteellisia osia, jotka vastaavat käsitteellisesti kiintolevyn hakemisto- tai kansiorakennetta. Kaikki virtuaalihakemistot ovat Web-sivuston kotihakemiston alihakemistoja. Niitä kutsutaan virtuaalihakemistoiksi, koska niiden rakenne ei aivan täysin vastaa kiintolevyllä olevaa hakemistorakennetta. Jokainen virtuaalihakemisto osoittaa kiintolevyllä olevaan hakemistoon, jossa on Web-sivuja, ActiveX-dokumentteja ja muita tiedostoja, jotka muodostavat virtuaalihakemiston *sisällön* (content). Näiden sisältöhakemistojen ei tarvitse olla edes samassa tietokoneessa Web-palvelimen kanssa.

Virtuaalihakemistojen URL:ssä käytetään UNIX-tyylistä jakoviivaa. Jos kirjoitat selaimen **Address**-ruutuun **e**simerkiksi seuraavan URL:n, selain avaa Microsoftin Web-palvelun visualc-virtuaalihakemiston:

```
http://www.microsoft.com/visualc/
```

Jos tarkkaa Web-sivua ei URL:ssä nimetä, kuten juuri nähtiin, Web-palvelin avaa määrätyn virtuaalihakemiston oletussivun.

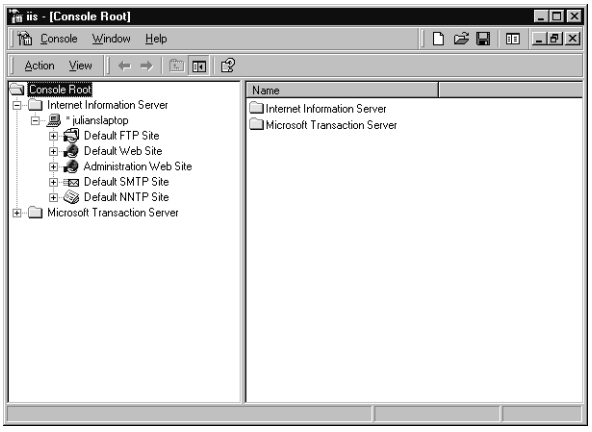
Sisällön jakelu Microsoft IIS 4.0 Web-palvelimella

Microsoft Internet Service Manager on IIS:n mukana asennettu apuohjelma, jonka avulla hallinnoidaan Web-palvelinta ja muita IIS:n tarjoamia Internet-palveluita, kuten sähköpostia ja FTP-palvelua. Käyttämällä Internet Service Manageria voit konfiguroida Web-palvelun sisältöä, turvallisuutta ja suorituskykyä.

Seuraavassa harjoituksessa luodaan Internet Service Manageria käyttämällä *docs*-virtuaalihakemisto Web-palveluun. Hello.ads ActiveX-dokumentti sijoitetaan docs-vituaalihakemistoon.

► **Hello.ads ActiveX-dokumentin jakelu**

1. Luo Resurssienhallintaa käyttämällä uusi kansio \InetPub\WWWRoot-kansion alle ja anna sille nimi **MyActiveXDocs**. Kopioi Hello.ads-tiedosto tähän kansioon.
2. Valitse **Käynnistä**-valikosta **Ohjelmat**, **NT 4.0 Option Pack**, **Microsoft Internet Information Server**, ja napauta **Internet Service Manager**.
3. Avaa vasemmasta osasta **Internet Information Server** -yksikkö. Avaa kohta, joka on nimetty samalla nimellä kuin oma tietokeneesi niin, että ikkuna näyttää samanlaiselta kuin kuvassa 12.17.



Kuva 12.17 Internet Service Manager

4. Napauta hiiren kakkospainikkeella **Default Web Site**. Valitse **New**-alivalikosta **Virtual Directory**.
5. Kirjoita New Virtual Directory Wizardissa nimeksi **docs** ja napauta **Next**.
6. Napauta **Browse** ja etsi \InetPub\WWWRoot\MyActiveXDocs-kansio. Määrittele, että tämä kansio sisältää docs-virtuaalihakemiston sisällön napauttamalla **OK**. Napauta **Next**.
7. Hyväksy velhon viimeisen sivun oletusasetukset napauttamalla **Finish**. Jos napautat vasemmassa osassa kohtaa **Default Web Site**, näet oikealla docs-virtuaalihakemiston.
8. Sulje Internet Service Manager. Avaa Internet Explorer ja kirjoita seuraava URL-osoiteruutuun:

   http://[*your computer name*]/docs/hello.ads

Hello.ads-tiedoston tulisi avautua selaimeen kuten kuvassa 12.16.

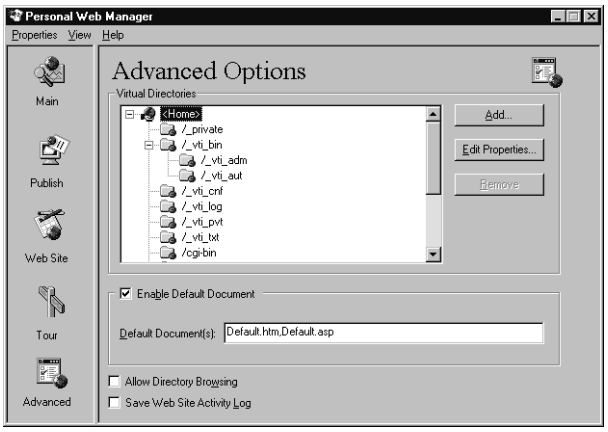
Sisällön jakelu Microsoft IIS Personal Web Serverillä

PWS:ssä on Personal Web Manager -toiminto, jota käytetään Web-sivun hallinnointiin.

Seuraavassa harjoituksessa käytetään Personal Web Manageria *docs*-virtuaalihakemiston luomiseen oletus-Web-palveluun. Hello.ads ActiveX-dokumentti sijoitetaan docs-virtuaalihakemistoon.

► **Hello.ads ActiveX-dokumentin jakelu**

1. Tee Resurssienhallintaa käyttäen uusi kansio \InetPub\WWWRoot-kansioon ja anna sille nimeksi **MyActiveXDocs**. Kopioi Hello.ads-tiedosto tähän kansioon.
2. Käynnistä Personal Web Manager valitsemalla **Käynnistä**-valikosta **Programs** ja napauttamalla sitten **Microsoft Personal Web Server**.
3. Personal Web Managerin **Main**-sivu avautuu. Napauta vasemmasta osasta **Advanced** niin, että sovellus näyttää samanlaiselta kuin kuvassa 12.18.



Kuva 12.18 Personal Web Manager

4. Kotihakemiston ollessa valittuna kuten kuvassa 12.18 napauta **Add**.
5. Add Directory Wizard avautuu. Kirjoita **Alias**-ruutuun **docs**. Napauta **Browse** ja etsi \InetPub\WWWRoot\MyActiveXDocs-kansio. Lisää tämä kansio **Directory**-ruutuun napauttamalla **OK**. Luo virtuaalihakemisto napauttamalla **OK**.
6. Varmista, että virtuaalihakemisto on lisätty kotihakemiston alle Virtual Directories -ikkunassa. Sulje Personal Web Manager.
7. Käynnistä Internet Explorer ja kirjoita seuraava URL osoiteruutuun:

   http://[*your computer name*]/docs/hello.ads

Hello.ads-tiedoston pitäisi avautua selaimeen kuten kuvassa 12.16.

Oppitunnin yhteenveto

ActiveX-dokumentit luodaan ActiveX-palvelinsovelluksilla. ActiveX-dokumentteja voidaan katsella ja muokata sillä sovelluksella, joka ne on luonut tai millä tahansa ActiveX-dokumenttisäilösovelluksella. Kun säilösovellus lataa ActiveX-dokumentin, se lataa siihen liittyvän palvelinsovelluksen resurssit. Palvelinsovelluksen valikot ja työkalurivit yhdistetään säilösovelluksen valikoihin ja työkaluriveihin, jolloin palvelinsovelluksen toiminnot ovat käytettävissä myös säilösovelluksessa.

Internet Explorer on eniten käytetty ActiveX-dokumenttisäilösovellus. Internet Exploreria käytetään usein Web-sivuilla levitettävien ActiveX-dokumettien katselemiseen ja jakeluun. ActiveX-dokumenttien avulla Web-sivujen käyttöliittymä saadaan paljon monipuolisemmaksi kuin pelkällä HTML:llä.

Ennen kuin ActiveX-dokumentteja voidaan katsella Internetin kautta täytyy paikalliselle tietokoneelle asentaa vastaava ActiveX-dokumenttipalvelinsovellus. Näin ollen ActiveX-dokumenttien käyttö onnistuu parhaiten tilanteissa, joissa asiakastietokoneiden käyttöjärjetelmä ja asetukset voidaan ennalta määrätä, kuten esimerkiksi yrityksen tietokoneissa.

MFC AppWizard, joka toimitetaan Visual C++ 6.0:n osana sisältää toiminnon, jonka avulla ActiveX-palvelinsovelluksen tekeminen on helppoa.

Oppitunti 3: Web-palvelimen ohjelmointi

Oppitunnilla 2 tutustuit Microsoftin Web-palvelinsovelluksiin: IIS 4.0 ja PWS. Opit, kuinka niiden avulla järjestetään ja jaetaan sisältöä kuten HTML-sivuja ja ActiveX-dokumentteja Web-palvelussa.

Tällä oppitunnilla opit, kuinka kirjoitetaan ohjelmia, jotka laajentavat Microsoft Web serverin toiminnallisuutta käyttämällä ISAPIa. Tutustut kahden tyypisiin ohjelmiin: ISAPI *palvelinlaajennuksiin* (server extensions) ja ISAPI *suodattimiin* (filters), jotka molemmat toteutetaan DLL:inä. ISAPI-palvelinlaajennukset ovat ohjelmia, joita suoritetaan Internetpalvelinprosessissa. ISAPI suodattimet ottavat vastaan palvelimelle menevää ja palvelimelta tulevaa tietoa ja voivat näin hoitaa henkilökohtaisia kirjautumis-, salaus- ja muita vastaavia tehtäviä.

Tämän oppitunnin jälkeen:

- Tiedät, kuinka tehdään ISAPI-palvelinlaajennuksia MFC:llä.
- Tiedät, kuinka ISAPI palvelinlaajennus saadaan tuottamaan mukautettua sisältöä HTML-sivulta välitettyjen parametrien perusteella.
- Tiedät, kuinka tehdään ISAPI-suodatin MFC:llä.
- Tiedät, kuinka ISAPI-suodatin asennetaan.

Oppitunnin arvioitu kesto: 30 minuuttia

ISAPI-palvelinlaajenukset

ISAPI-palvelinlaajennus on palvelimella toimiva ohjelma, joka suoritetaan selaimen pyynnöstä. Palvelinlaajennukset toteutetaan DLL:inä, joita IIS lataa. Selain voi käynnistää ohjelman määrittämällä URL:ssä .dll-tiedoston nimen seuraavaan tapaan:

```
http://myserver/apps/charts.dll
```

ISAPI-palvelinlaajennus toteuttaa toimintoja, joita voidaan kutsua selaimista. Toimintoa kutsutaan lisäämällä toiminnon nimi URL:ään seuraavasti:

```
http://myserver/apps/charts.dll?GetChart
```

Tässä esimerkissä viitataan *charts.dll*-palvelinlaajennukseen ja kutsutaan **GetChart()**-funktiota. Kaikissa ISAPI-palvelinlaajennuksissa täytyy määritellä oletusfunktio, jota kutsutaan, mikäli käyttäjä ei määritä funktiota erikseen. Palvelinlaajennusfunktiot voivat ottaa vastaan parametrejä seuraavalla tavalla:

```
http://myserver/apps/charts.dll?GetChart?Fund=ARSC
```

ISAPI-palvelinlaajennus voidaan linkittää HTML-lomakkeeseen määrittämällä laajennuksen URL <FORM>-muotoilun *action*-attribuuttiin seuraavasti:

```
<FORM action="myextension.dll?GetColor" method=POST>
<!— form controls go here —>
</FORM>
```

Lomakkeen kontrolleihin kirjoitetut tiedot välitetään automaattisesti ISAPI-palvelinlaajennukselle parametreinä, kun käyttäjä lähettää lomakkeen. ISAPI-palvelinlaajennus toimii tyypillisesti niin, että se lähettää selaimelle näihin parametritietoihin perustuvan mukautetun HTML-sivun.

ISAPI-palvelinlaajennukset perustuvat Common Gateway Interface (CGI) -standardiin. CGI on HTTP:n osa, jonka avulla selainohjelmat voivat toimia vuorovaikutuksessa skriptien tai erillisten Web-palvelimella olevien sovellusten kanssa. Muuttamatta HTTP/CGI-määrityksiä, Microsoft kehitti IIS:ään ominaisuuden, joka antaa kaikille selaimille mahdollisuuden ladata ja suorittaa palvelinlaajennus DLL:ä. Koska DLL:t toimivat siinä prosessissa, joka ne on ladannut, palvelinlaajennukset ovat nopeampia kuin skriptit, jotka täytyy mahdollisesti ladata erilliseen ohjelmaan suorittamista varten.

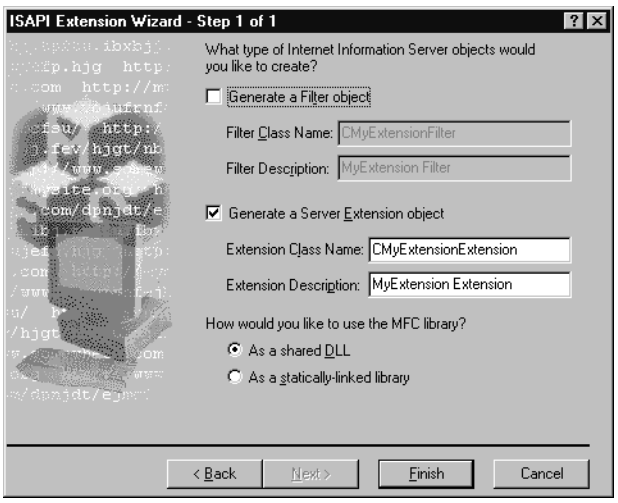
CGI siirtää ohjelmoinnin vaivan palvelimelle. CGI-parameterejä käyttämällä selain lähettää pieniä määriä informaatiota palvelimelle. Palvelin voi käyttää tätä tietoa aivan kuten tahtoo, mukaanlukien tietokannan käsittely, kuvien luominen ja oheislaitteiden käyttäminen. Palvelin lähettää takaisin selaimelle tiedoston (HTML tai muun), joka voidaan hakea palvelimen kiintolevyltä tai muodostaa ohjelmallisesti.

ISAPI-palvelinlaajennusten luominen MFC:llä

Jotta näkisimme kuinka perus-ISAPI-palvelinlaajennus toimii, luomme nyt sellaisen MFC:n ISAPI Extension Wizardia käyttäen.

► **MyExtension ISAPI-palvelinlaajennus-DLL:n luominen**

1. Valitse Visual C++:n **File**-valikosta **New**. Valitse **New**-dialogin **Projects**-sivulta **ISAPI Extension Wizard**. Kirjoita projektin nimeksi **MyExtension** ja napauta **OK**.
2. ISAPI Extension Wizard avautuu kuten kuvassa 12.19.

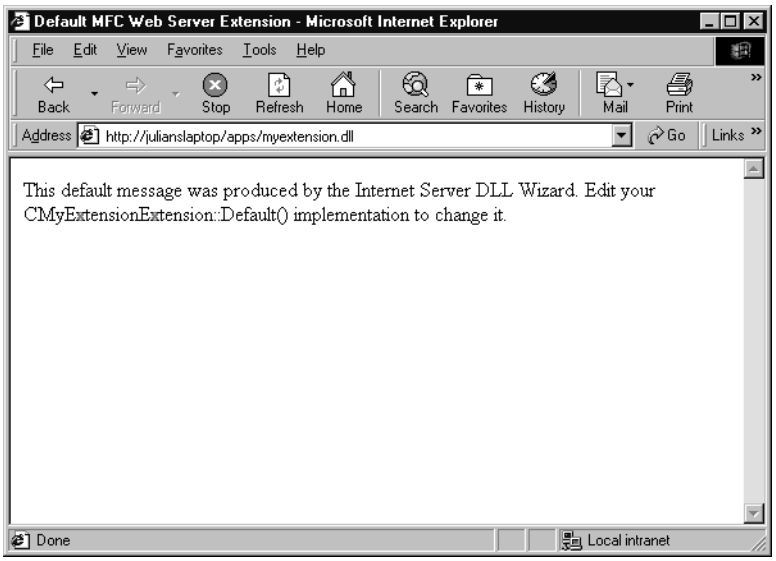


Kuva 12.19 ISAPI Extension Wizard

3. Hyväksy oletusasetukset napauttamalla **Finish** ja luo sitten projekti napauttamalla **OK**.
4. Luo MyExtension.dll ISAPI-palvelinlaajennus kääntämällä projekti.
5. Luo Resurssienhallintaa käyttäen \InetPub\WWWRoot-kansioon kansio **MyISExtensions**. Kopioi MyExtension.dll-tiedosto tähän hakemistoon.
6. Luo Web-palvelimesi hallintaohjelmalla virtuaalihakemisto sivuston kotihakemiston alle ja anna sen nimeksi **apps**. Määritä, että tämän kansion sisältö sijoitetaan C:\INetPub\WWWRoot\ MyISExtensions-kansioon. Varmista, että tähän kansioon on **Suoritus** (Execute) -oikeudet.
7. Käynnistä Internet Explorer ja kirjoita seuraava URL **Address**-ruutuun:

   http://[*your computer name*]/apps/myextension.dll

   Tämä pyyntö kutsuu MyExtension.dll:n oletusfunktiota. Tuloksen tulisi näyttää samanlaiselta kuin kuvassa 12.20.



Kuva 12.20 Internet Explorer 5 näyttää MyExtension.dll:n tulosteen

MyExtension-koodiin tutustuminen

Jos tutkit MyExtension-projektin koodia näet, että ISAPI Extension Wizard on luonut yhden **CMyExtensionExtension**-nimisen luokan. Tämä luokka on periytetty MFC:n kantaluokasta **CHttpServer**. Voit lisätä tähän luokkaan funktioita, jotka toteuttavat DLL:n julkistamat toiminnot.

MFC luo DLL:n toimintojen määrittämistä ja niiden **CHttpServer**istä periytetyn luokan funktioille ohjaamista varten *parse map* -nimisen rakenteen. Parse map määritellään header-tiedostossa **DECLARE_PARSE_MAP**-makron mukana ja toteutetaan **BEGIN_PARSE_MAP** ja **END_PARSE_MAP** makroissa, kuten seuraavasta MyExtension.cpp-tiedostosta otetusta esimerkistä nähdään:

```
BEGIN_PARSE_MAP(CMyExtensionExtension, CHttpServer)
    // TODO: insert your ON_PARSE_COMMAND() and
    // ON_PARSE_COMMAND_PARAMS() here to hook up your commands.
    // For example:

    ON_PARSE_COMMAND(Default, CMyExtensionExtension, ITS_EMPTY)
    DEFAULT_PARSE_COMMAND(Default, CMyExtensionExtension)
END_PARSE_MAP(CMyExtensionExtension)
```

ISAPI Extension Wizard on lisännyt **DEFAULT_PARSE COMMAND** makron, joka määrittelee oletusfunktioksi funktion **Default()**. Wizardin tekemä yksinkertainen funktio löytyy, kun tiedostoa selataan eteen päin:

```
void CMyExtensionExtension::Default(CHttpServerContext* pCtxt)
{
    StartContent(pCtxt);
    WriteTitle(pCtxt);

    *pCtxt << _T("This default message was produced by the Internet");
    *pCtxt << _T("
        Server DLL Wizard. Edit your CMyExtensionExtension::Default()");
    *pCtxt << _T(" implementation to change it.\r\n");

    EndContent(pCtxt);
}
```

Tämän funktion tuotos nähdään kuvassa 12.20.

Default()-funktio, kuten kaikki ISAPI:n laajennusfunktiot, saa **CHttpServerContext**-objektin osoitteen. **CHttpServer** luo yhden **CHttpServerContext**-objektin jokaista HTTP asiakas/palvelin-siirtoa kohden. Kun palvelinlaajennus-DLL käsittelee pyyntöä, se käyttää **CHttpServerContext**-jäsenfunktiota suorittaessaan tehtäviä kuten hakiessaan HTTP-asiakkaan palvelupyynön yksityiskohtia otsikosta (käyttäen funktiota **CHttpServerContext::GetServerVariable()**), tai sijoittaessaan HTML-tekstiä selaimelle lähetettävään vastaustiedostoon (käyttämällä ylikuormitettua **<<**-operaattoria).

Palvelinlaajennuksen **CHttpServer**-objekti luo **CHttpServerContext**-objektin jokaista asiakkaan pyyntöä kohden. Jokainen objekti luodaan omaan säikeeseensä, mikä mahdollistaa useat yhtäaikaiset **CHttpServer**-objektin kutsut eri asiakkailta. Luotaessa synkronointia globaaleille muuttujille tai mille tahansa **CHttpServer**-luokan jäsenille täytyy olla varovainen.

Funktion lisääminen palvelinlaajennukseen

Seuraavaksi lisätään yksinkertainen **GetColor()-niminen funktio** ISAPI-palvelinlaajennukseen. **GetColor()** ottaa parametrinä yhden HTML:n värien nimistä ja luo sivun, jonka taustavärinä on parametrina määritelty väri.

► **GetColor()-funktion lisääminen**

1. Lisää seuraavat rivit MyExtension.cpp-tiedoston Parse Mapiin:

```
ON_PARSE_COMMAND(GetColor, CMyExtensionExtension, ITS_PSTR)
ON_PARSE_COMMAND_PARAMS("color")
```

ON_PARSE_COMMAND makro määrittelee funktion nimen — luokan nimen, johon toiminto ohjataan. Funktion vastaan ottamat parametrit määritellään kolmantena (ja seuraavina) **ON_PARSE_COMMAND** makron parametrinä. Parametrin tyypit määritellään **ON_PARSE_COMMAND _PARAMS**-makron avulla. Edellä nähdyt koodirivit määrittävät, että **GetColor()**-funktiolla on yksi merkkijonoparametri *color*.

2. Lisää seuraavan tunnisteen omaava funktio ClassViewin **Add Member Function** -toiminnolla **CMyExtensionExtension**-luokkaan:

```
void GetColor(CHttpServerContext *pCtxt, LPCTSTR pstrColor)
```

3. Lisää seuraava funktion toteutus:

```
void CMyExtensionExtension::GetColor(CHttpServerContext *pCtxt, LPCTSTR pstrColor)
{
    StartContent(pCtxt);
    WriteTitle(pCtxt);

    *pCtxt << _T("You have chosen the ");
    *pCtxt << pstrColor << _T(" page.\r\n");
    *pCtxt << _T("<SCRIPT language=\"JavaScript\">
        document.bgColor = \"");
    *pCtxt << pstrColor << _T("\" </SCRIPT>\r\n");

    EndContent(pCtxt);

}
```

4. Käännä ja käynnistä MyExtension-projekti. Uusi DLL täytyy kopioida \INetPub\WWWRoot\MyISExtensions -kansioon niin, että se korvaa edellisen version. Ennen kuin tämä voidaan tehdä, täytyy tietokoneen WWW-palvelu pysäyttää. Tämä johtuu siitä, että ISAPI-palvelinlaajennukset ladataan IIS-prosessiin, kun asiakas pyytää niitä ensimmäistä kertaa ja ne pysyvät siellä, kunnes WWW-palvelu pysäytetään. Helpoin tapa WWW-palvelun pysäyttämiseen on avata komentokehote ja kirjoittaa seuraavasti:

```
net stop w3svc
```

5. Kun olet kopioinut DLL:n, käynnistä palvelu uudelleen kirjoittamalla seuraavasti:

```
net start w3svc
```

Uusi DLL ladataan ensimmäisen asiakkaan palvelupyynnön seurauksena.

Windows 95 ja Windows 98 -ympäristöissä Web-palvelu käynnistetään Personal Web Managerin **Properties**-valikosta. Päästäksesi käsittelemään

DLL:ää, sinun täytyy mahdollisesti käynnistää tietokone uudelleen palvelun pysäyttämisen jälkeen.

Voit kokeilla laajennuksen toimintaa seuraavan *Form.htm*-nimisen HTML-tiedoston avulla. Sivulla on yksinkertainen lomake, jonka luetteloruudusta käyttäjä voi valita värin. Kun käyttäjä napautaa **Submit**-painiketta, palvelimelta lähetetään selaimelle valitun värin värinen sivu.

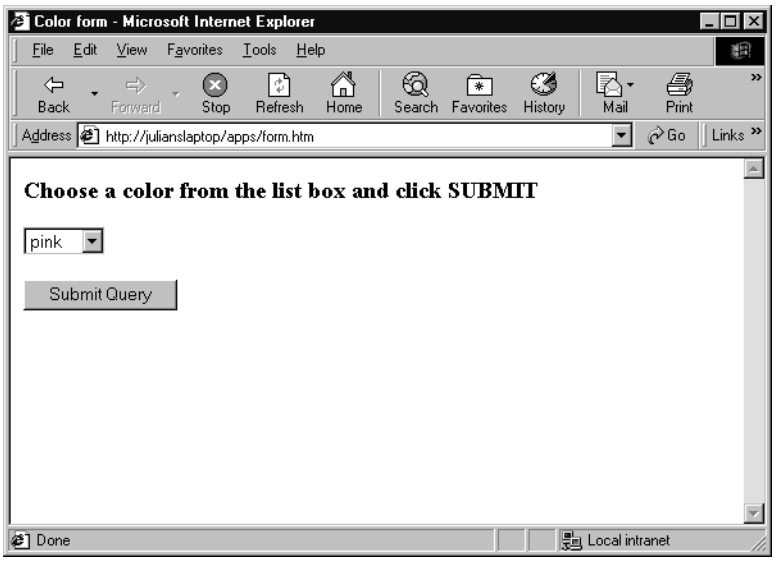
```
<!— Form.htm —>
<HTML>
<HEAD>
    <TITLE>
        Color Form
    </TITLE>
</HEAD>
<BODY>
    <H3> Choose a color from the list box and click SUBMIT</H3>

    <FORM action="myextension.dll?GetColor" method=POST>
    <SELECT name="color">
        <option> pink
        <option> green
        <option> yellow
        <option> blue
    </SELECT>
    <P>
    <input type="submit">
    </FORM>
</BODY>
</HTML>
```

Form.htm on Chapter 12\Exercises -kansiossa. Kopioi Form.htm kansioon \INetPub\WWWRoot\MyISExtensions.

6. Käynnistä Internet Explorer ja kirjoita seuraava URL osoiteruutuun:

   http://[*your computer name*]/apps/form.htm

   Kuvassa 12.21 näkyvä sivu avautuu.



Kuva 12.21 Form.htm avattuna Internet Explorer 5:llä

7. Valitse väri luetteloruudusta ja napauta **Submit Query**. Selaimeen avautuu sivu, jonka tausta on valitun värinen.

ISAPI-suodattimet

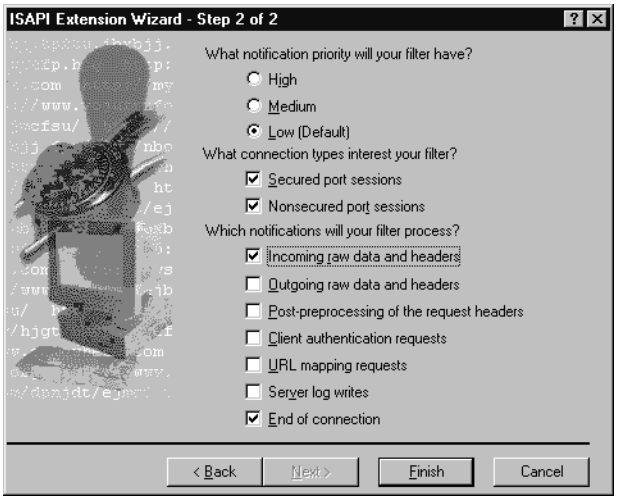
ISAPI-suodattimet sijoittuvat verkkoyhteydessä asiakkaan ja HTTP-palvelimen väliin. Niiden avulla voidaan Internet-palveluun lisätä mukautettuja ominaisuuksia kuten tiettyjen HTTP-pyyntöjen kirjaaminen, asiakaskohtaisesti mukautettu pakkaus tai salaus, sekä vaihtoehtoiset tunnistusmenetelmät. Suodattimet toteutetaan DLL:inä, jotka ladataan, kun WWW-palvelu käynnistetään. Voit lisätä suodattimia IIS 4.0 Webpalveluihin Internet Service Managerilla. Personal Web Server ei tue ISAPI-suodattimia.

Suodattimelle voidaan ilmoittaa valituista palvelimen tapahtumista. Palvelin voi ilmoittaa suodattimelle esimerkiksi käsitellessään raakadataa asiakkaan pyynnöstä, lähettäessään tietoja asiakkaalle, kirjoitettuaan logiin tai suljettuaan yhteyden. Ilmoitusta käsitellessään suodatin pääsee käsiksi palvelimella oleviin ilmoitukseen liittyviin tietoihin. Suodattimelle voidaan ilmoittaa esimerkiksi, kun palvelin lähettää asiakkaalle käsittelemätöntä dataa, jolloin suodatin voi muokata tietoja esimerkiksi pakkaamalla tai salaamalla tiedot.

Seuraavissa harjoituksissa havainnollistetaan yksinkertaisen ISAPI-suodattimen luomista ISAPI Extension Wizardilla.

► **MyFilter ISAPI-suodattimen luominen**

1. Valitse **New**-dialogin **Projects**-välilehdeltä **ISAPI Extension Wizard**. Kirjoita projektin nimeksi **MyFilter** ja napauta **OK**.
2. Poista ISAPI Extension Wizardin ensimmäisessä vaiheessa valinta **Generate a Sever Extension object** ja valitse **Generate a Filter object**.
3. Valitse ISAPI Extension Wizardin toisessa vaiheessa **Incoming raw data and headers**. Jätä **End of connection** valituksi niin, että dialogi näyttää samanlaiselta kuin kuvassa 12.22.



Kuva 12.22 ISAPI-suodattimen luominen

4. Luo MyFilter-projekti napauttamalla ensin **Finish** ja sitten **OK**.

Jos tarkastelet luotua koodia huomaat, että ISAPI Extension Wizard on luonut yhden **CMyFilterFilter**-nimisen luokan, joka on periytetty MFC:n luokasta **CHttpFilter**. Tähän luokkaan on lisätty käsittelijä jokaiselle ISAPI Extension Wizardin vaiheessa 2 määritellylle ilmoitukselle, kuten seuraavassa juuri luodussa **CMyFilterFilter**-luokan määrittelyssä nähdään:

```
virtual DWORD OnReadRawData(CHttpFilterContext* pCtxt,
    PHTTP_FILTER_RAW_DATA  pRawData);
virtual DWORD OnEndOfNetSession(CHttpFilterContext* pCtxt);
```

Nämä funktiot ylikirjoitetaan luokassa **CHttpFilter**. Wizard tekee .cpp-tiedostoon tynkäfunktiot. Suodattimen toiminta määritellään muuttamalla näitä funktioita.

Luokassa on toteutettu myös toinen ylikuormitettu funktio **GetFilterVersion()**. Palvelin selvittää suodattimen käsittelemät ilmoitukset kutsumalla tätä luokkaa. Suodatusprosessin tehostamiseksi palvelin lähettää suodattimelle ilmoituksen

vain niistä tapahtumista, jotka on määritelty tässä funktiossa. Seuraava **MyFilter ::GetFilterVersion()** funktiosta poimittu rivi näyttää, kuinka tapahtuma määritellään:

```
pVer->dwFlags |= SF_NOTIFY_ORDER_LOW | SF_NOTIFY_SECURE_PORT |
    SF_NOTIFY_NONSECURE_PORT | SF_NOTIFY_READ_RAW_DATA |
    SF_NOTIFY_END_OF_NET_SESSION;
```

Huomaa, että jos lisäät uusia käsittelijöitä uusille ilmoituksille, sinun täytyy myös lisätä vastaava lippu tälle riville. Vaikka ilmoitusten käsittelijöitä voidaan lisätä ClassWizardilla, se ei päivitä **GetFilterVersion()**-funktiota, joten sen muuttaminen täytyy muistaa tehdä manuaalisesti.

Ilmoitusten käsittelijät saavat tietorakenteen, joka sisältää käsiteltävää tapahtumaa koskevia tietoja. Ne saavat myös osoittimen **CHttpFilterContext**-objektiin, joka kuten **CHttpServerContext**-objektikin, sisältää tiedot senhetkisestä asiakasyhteydestä ja sisältää metodit, jotka tarvitaan haettaessa tietoja yhteydestä tai kirjoitettaessa tietoja asiakkaalle menevään vastaukseen.

Seuraavassa harjoituksessa tehdään yksinkertainen **MyFilter:: OnReadRawData()**-funktion toteutus. Siinä käytetään **CHttpFilterContext:: GetServerData()** funktiota etäpalvelimen IP-osoitteen selvittämiseen ja kirjoitetaan se sitten lokitiedostoon.

► MyFilter::OnReadRawData()-funktion toteutus

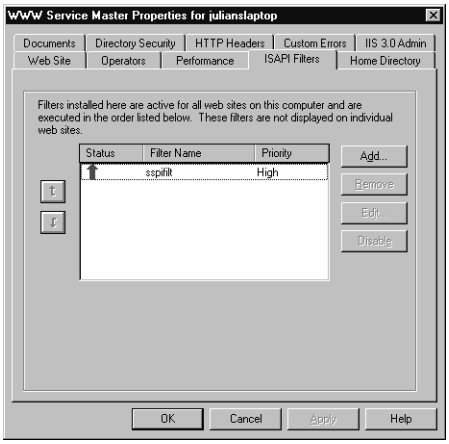
1. Etsi **MyFilter::OnReadRawData()**-funktio ja korvaa se seuraavalla toteutuksella:

```
DWORD CMyFilterFilter::OnReadRawData(CHttpFilterContext* pCtxt,
PHTTP_FILTER_RAW_DATA  pRawData)
{
    char pchVar[64];
    DWORD dwSize = 64;

    CStdioFile logfile("C:\\iislog.txt",
        CFile::modeCreate | CFile::modeWrite);

    BOOL bRet = pCtxt->GetServerVariable(
        "    REMOTE_HOST", pchVar, &dwSize);

    if(bRet)
        logfile.Write(pchVar, dwSize);

    logfile.Close();

    return  SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

2. Käännä MyFilter-projekti. Kopioi MyFilter.dll-tiedosto kansioon c:\Winnt\System32\inetsrv.

► **Suodattimen asennus IIS 4.0 Serveriin**

1. Avaa Internet Service Manager. Napauta hiiren kakkospainikkeella paikallisen palvelimen kuvaketta (kuvakkeessa näkyy tietokoneesi nimi) ja valitse **Properties**.
2. Varmista **Master Properties** -ruudusta, että **WWW Service** on valittu ja napauta **Edit**.
3. Valitse **WWW Service Master Properties** -dialogista **ISAPI Filters** -välilehti. Näet välilehden kuvassa 12.23.



Kuva 12.23 ISAPI-suodattimen asentaminen Web-palvelimeen

4. Napauta **Add**. Kirjoita suodattimen nimeksi **MyFilter**. Käytä **Browse**-painiketta ja etsi MyFilter.dll-tiedosto c:\Winnt\System32\inetsrv-kansiosta ja kirjoita tiedoston koko polku **Executable**-ruutuun.
5. Sulje **WWW Service Master Properties** -dialogi napauttamalla **OK**. Sulje sen jälkeen Server Properties napauttamalla uudelleen **OK**.
6. Avaa järjestelmän komentokehote ja kirjoita:

```
net stop w3svc
```

ja tämän jälkeen:

```
net start w3svc
```

7. Avaa Internet Explorer ja avaa sivu paikalliselta palvelimelta — esimerkiksi http://[*your computer name*]/apps/form.htm, jota käytetiin edellisessä harjoituksessa.
8. Sulje Internet Explorer. Jos katsot c:\-asemasi juurihakemistoon, näet juuri luodun *iislog.txt*-tiedoston. Avaa tiedosto Muistioon kaksoisnapauttamalla. Näet, että koeneesi IP-osoite on lisätty tähän tiedostoon.

Oppitunnin yhteenveto

ISAPI:n avulla voit kirjoittaa ohjelmia, jotka laajentavat IIS 4.0:n tai PWS:n toiminnallisuutta.

On olemassa kahdenlaisia ISAPI-ohjelmia: palvelinlaajennuksia ja suodattimia, jotka molemmat toteutetaan DLL:inä. ISAPI-palvelinlaajennukset ovat käännettyjä palvelinosan ohjelmia, joita Internet-asiakasohjelma voi käynnistää. ISAPI-suodattimet käsittelevät palvelimelta tulevaa ja palvelimelle menevää tietoa ja voivat näin suorittaa erilaisia kirjautumistoimia, salata tietoja ja muita tehtäviä.

ISAPI-palvelinlaajennukset perustuvat CGI-standardiin ja ne voidaan käynnistää määrittelemällä DLL:n nimi URL:ssä. Palvelinlaajennukset julkistavat funktioita, jotka voivat ottaa parametrejä. Funktion nimi ja parametrit voidaan määrittää osana URL:iä tai laajennus voidaan yhdistää HTML-lomakkeeseen, jolloin se saa parametrinsä lomakkeen kontrolleista. Voit käyttää ISAPI Extension Wizardia luodessasi **CHttpServer** ja **CHttpServerContext** -luokkiin perustuvia laajennusprojekteja.

ISAPI suodattimet sijoittuvat asiakkaan ja HTTP-palvelimen väliseen verkkoyhteyteen ja lisäävät Internet-palveluun mukautettuja toimintojaan kuten mukautetun kirjautumisen, salauksen tai pakkauksen. Suodattimet toteutetaan DLL:inä, jotka ladataan samalla, kun WWW-palvelu käynnistetään. Voit yhdistää suodattimia IIS 4.0 -sivustoihin käyttämällä Internet Service Manageria.

Suodattimet saavat ilmoituksia, kun tiettyjä palvelintapahtumia syntyy. Käsitellessään näitä ilmoituksia suodatin saa käyttöönsä ilmoitukseen liittyvät palvelimen tiedot. Voit käyttää ISAPI Extension Wizardia luodessasi ISAPI-suodattimia, jotka perustuvat **CHttpFilter** ja **CHttpFilterContext** -luokkiin.

Laboratorio 12: STUpload muuttaminen ActiveX-dokumenttipalvelimeksi

Tässä laboratoriossa nähdään, kuinka täysin ominaisuuksin varustettu MFC-sovellus muutetaan helposti ActiveX-dokumenttipalvelimeksi. STUpload-sovelluksen toiminnallisuus saadaan kaikkien Internet Explorerin käyttäjien saataville niin, että he voivat katsella Web-palvelimella olevia STUpload-dokumentteja.

Koska esimerkit on haluttu pitää mahdollisimman suoraviivaisina, STUpload-projekti, jota olet käsitellyt aiemmin tässä kirjassa, ei sisällä tukea ActiveX-doku-menttipalvelintoiminnoille. Kansiossa Chapter 12\Partial on STUpload-projektin kopio, johon on liitetty tuki ActiveX-dokumenttipalvelimena toimimiselle. Kuten muistat, ActiveX-dokumenttipalvelintuki lisätään tekemällä **Full-server** ja **Active document server** -valinnat MFC AppWizardin vaiheesa 3. Käytä tämän projektin kopioita lähtökohtana tässä laboratoriossa.

► **ActiveX-dokumenttipalvelinluokkien tutkiminen**

1. Avaa Chapter 12\Partial\STUpload-projektin kopio. Jos katselet ClassViewissä näkyviä luokkia, huomaat, että projektin ActiveX-dokumenttipalvelinversiossa on muutamia lisäluokkia. **CInPlaceFrame**-luokka edustaa kehysikkunaa, kun dokumettia käsitellään “paikallaan” — eli kun sitä käsitellään säilössä. **CSTUploadServerItem**-luokka sisältää säilön palvelin rajapinnan.
2. Kaksoisnapauta **CSTUploadDoc**-kohtaa. Tutki luokan määrittelyä ja huomaa, että se on periytetty luokasta **COleServerDoc**.

Seuraavaksi muokataan **IDR_SRVR_INPLACE**-valikkoa ja työkaluriviä, jotta STUploadin valikkokomennot olisivat käytettävissä ActiveX-dokumenttisäilön valikoista.

► **IDR_SRVR_INPLACE-valikon muokkaaminen**

1. Kaksoisnapauta ResourceViewissä **IDR_SRVR_INPLACE**-valikkoa. Valitse **Edit**-alivalikko ja poista se painamalla **DEL**. Vahvista poisto napauttamalla **OK**.
2. Kaksoisnapauta **IDR_MAINFRAME**-valikkoa. Ota molemmat valikot kerralla näkyviin valitsemalla **Window**-valikosta **Tile Horizontally**.
3. Kopioi **Data**-alivalikko **IDR_MAINFRAME**-valikosta **Edit**-alivalikon entiselle paikalle **IDR_SRVR_INPLACE**-valikkoon. Tee kopiointi raahaamalla CTRL-painike pohjaan painettuna.
4. Poista **IDR_SRVR_INPLACE**-valikosta **Data**-alivalikon komento **Import**. Sulje molemmat menueditori ikkunat.

Import-komento poistettiin, koska STUpload-sovelluksen ActiveX-dokumenttiosaa käytetään vain valmiiden Web-sivuilta jaettujen dokumenttien katseluun ja lataamiseen. Dokumenttien luomista Webin kautta ei tehdä mahdolliseksi.

► **IDR_SRVR_INPLACE-työkalurivin muokkaaminen**

1. Kaksoisnapauta ResourceViewissä **IDR_SRVR_INPLACE**-työkaluriviä. Poista **Cut**, **Copy** ja **Paste** -painikkeet vetämällä ne ulos työkaluriviltä.
2. Kaksoisnapauta **IDR_MAINFRAME** työkaluriviä. Valitse **Window**-valikosta **Tile Horizontally**.
3. Kopioi painikkeet **ID_DATA_UPLOAD** ja **ID_DATA_QUERYDATABASE** valikosta **IDR_MAINFRAME** valikkoon **IDR_SRVR_ INPLACE**. Tee kopiointi raahaamalla pitäen samalla CTRL-näppäintä pohjassa. Sulje molemmat editori-ikkunat.

Käännä STUpload-projekti. Kun käännös on tehty onnistuneesti, tee seuraavat vaiheet, joilla tutkitaan STUpload-sovelluksen ActiveX-dokumenttipalvelintoimintoja.

► **STUpload ActiveX-dokumentin luominen ja jakelu**

1. Käynnistä STUpload painamalla CTRL+F5. Lataa Ch12Test.dat-tiedosto Chapter 12\Data-kansiosta käyttämällä **Data**-valikon **Import**-komentoa. Valitse tutkittava osake **Select Fund** ikkunasta.
2. Sulje STUpload ja tallenna samalla muutokset nimellä **ADTest.stu**.
3. Laita **ADTest.stu**-dokumentti jakeluun aiemmin tässä luvussa luomaasi **Docs**-virtuaalihakemistoon.

► **STUpload ActiveX-dokumenttipalvelintoimintojen kokeileminen**

1. Avaa Internet Explorer ja kirjoita seuraava URL **Address**-ruutuun:

   http://[*your computer name*]/docs/ADTest.stu

   ADTest.stu-tiedoston pitäisi avautua selaimeen. Jos koneellasi ei ole Web-palvelua asennettuna, voit simuloida tätä toimintoa raahaamalla ADTest.stu-tiedoston Resurssienhallinnasta selaimeen.
2. Valitse **Data**-valikosta **Upload**-komento. Tallenna tiedosto tietokantaan napauttamalla **OK**.
3. Kun lataus on suoritettu, valitse **Query Database** -komento **Data**-valikosta. Varmista, että tallettamasi muutokset ovat tietokannassa.
4. Sulje Internet Explorer.

Kertaus

1. Mikä on suositeltavin tapa selaintyylisen sovelluksen tekemiseen?

2. Kuinka ATL HTML -kontrollissa määritellään kontrollin esittämän HTML-sivun lähdekoodissa käytettävissä olevat metodit?

3. Kuinka Microsoft Scriptlet Component käsittelee skriptletin mukautetut tapahtumat?

4. Kuinka voit tallentaa Web-palvelimelta haettuun ActiveX-dokumenttiin tehdyt muutokset?

5. Kuinka määritellään mitkä ActiveX-palvelinsovelluksen valikkokomennoista näkyvät ActiveX-dokumenttisäilössä, kun se lataa ActiveX-dokumentin?

6. Kuinka ISAPI-palvelin laajennuksen oletus toiminto määritellään?

7. Kuinka ISAPI-palvelinlaajennuksen ja ISAPI-suodattimen lataaminen eroavat?