Lab 11.2: Using the HTTP WinInet Classes

Objectives

After completing this lab, you will be able to use the WinInet classes to:

- Create an Internet session.
- Get an HTTP connection.
- Open a request to an HTTP server.
- Send a request to an HTTP server.
- Read and display the data retrieved from the HTTP server.

Prerequisites

You should have completed Chapter 11 through the section titled "Writing HTTP Applications."

Lab Setup

This demonstration shows what you will accomplish during the lab.

Estimated time to complete this lab: **30 minutes**.

Exercises

The following exercise provides practice working with the concepts and techniques covered in this chapter.

Exercise 1: Creating an HTTP Application

In this exercise, you will add the code to:

- Create an Internet session.
- Obtain an HTTP connection.
- Open a request to retrieve a file.
- Send the request to the HTTP server.
- Read the header and data received from the server and display it in the application's view.

Before you start this lab, you should have installed Internet Explorer. You also must have an account with an Internet service provider or access to the Internet via a corporate firewall.

Alternatively, you can test these labs on a standalone computer. To do so, ensure that the Microsoft Personal Web Server (PWS)software is installed on your computer and that the Web server is properly configured and started. For more information on installing and configuring PWS, see the section Microsoft Personal Web Server in Chapter 11 of this course.

Copy the contents of \Labs\C11\Lab02\Baseline to your working directory.

The completed code for these exercises is in \Labs\C11\Lab02\Xxx, where Xxx is the exercise number.

Exercise 1: Creating an HTTP Application

The purpose of this lab is to create an application that accepts a URL from a user and then sequentially executes each part of a client HTTP transaction. Once the data is received from the HTTP server, the client application displays the data in the application view.

In this exercise, you will add the code to:

- Create an Internet session.
- Obtain an HTTP connection.

- Open a request to retrieve a file.
- Send the request to the HTTP server.
- Read the header and data received from the server and display it in the application's view.

About the Baseline Application

The baseline application's view class already has menu command handlers for executing each part of a client HTTP transaction:

- **CIC_2View::OnInternetObtainUrl**
- **CIC_2View::OnInternetCreateSession**
- **CIC_2View::OnInternetGetConnection**
- **CIC_2View::OnInternetOpenRequest**
- **CIC_2View::OnInternetSendRequest**
- **CIC_2View::OnInternetReadInformation**

Only **OnInternetObtainUrl** contains implementation code. This handler invokes a dialog box that accepts the URL the user entered. You will implement the remaining handlers in this lab.

The application is designed so that when the user enters a URL, the first menu item is enabled. The remaining menu items are enabled, in top-down order of their appearance on the menu, only after the previous menu item has been selected. To control the enabling of menu items, the application uses a state variable, m_processState. Each handler simply increments the value of this variable before exiting. Each menu item, using command updating, checks the value of m_processState to determine whether it is enabled.

Alternatively, a user can select the menu item "Obtain URL" followed by the menu item "All The Above" to complete all parts of a client HTTP transaction for the requested URL.

When the HTTP data is received, the headers as well as the body are added to a **CStringArray** object that is embedded in the application's **CDocument** object. The string array is then displayed to the application's view by forcing an update of the view.

➢ **Create an Internet Session**

1. Use Developer Studio to open the baseline application's Project Workspace, \Labs\C11\Lab02\Ex01\Baseline\Ic_2.Dsw.
2. Use the Developer Studio editor to open the file IC_2View.Cpp.
3. Add the following code to the body of the menu item handler **CIC_2View::OnInternetCreateSession**:

```
// Member function to close any previous session.
CloseSession();

// Create a new session.
m_pInternetSession = new CInternetSession;
ASSERT(m_pInternetSession != 0);

// Update the state of the menu items.
m_processState++;
```

Now that a session has been created, you can use the session's member function, **CInternetSession::GetHttpConnection**, to obtain an HTTP connection to the target HTTP server.

➢ **Obtain an HTTP connection to the target HTTP server**

1. Use the Developer Studio editor to open the file IC_2View.Cpp.
2. Add the following code to the body of the menu item handler **CIC_2View::OnInternetGetConnection**:

```
CIC_2Doc* pDoc = GetDocument();
```

```
    // Attempt to get an HTTP connection.
    try {
        m_pHttpConnection =
            m_pInternetSession->GetHttpConnection(pDoc->m_Server,
            pDoc->m_Port);
    }
    catch (CInternetException *e)
    {
        char buff[256];
        e->GetErrorMessage(buff,256);
        MessageBox(buff);
        e->Delete();
        return;
    }
    ASSERT(m_pHttpConnection != NULL);

    // Update the state of the menu items.
    m_processState++;
```

Now that a connection exists, the next step is to open an HTTP request specifying a GET command, which is used to retrieve a file. **CHttpConnection::OpenRequest** returns an Internet file handle that is used to formulate the request header, send the request, and then receive the server response. The **CDocument** object, m_Object, is a string containing the name of the file to retrieve from the server. If a file name was not specified in the URL the user entered, a default file will be returned.

➢ **Open an HTTP Request**

1. Use the Developer Studio editor to open the file IC_2View.Cpp.
2. Add the following code to the body of the menu item handler **CIC_2View::OnInternetOpenRequest**:

```
// Set some request flags.
    DWORD dwHttpRequestFlags = INTERNET_FLAG_EXISTING_CONNECT |
                               INTERNET_FLAG_NO_AUTO_REDIRECT;

    CIC_2Doc* pDoc = GetDocument();

    // Open a GET request and get an Internet File handle to communicate
through.
    m_pHttpFile = m_pHttpConnection-
>OpenRequest(CHttpConnection::HTTP_VERB_GET,
                               pDoc->m_Object,
                               NULL, 1, NULL, NULL, dwHttpRequestFlags);

    if (NULL == m_pHttpFile)
    {
        CString s;
        s.Format ("An error occurred opening the request");
        MessageBox(s);
        return;
    }

    // Update the state of the menu items.
    m_processState++;
```

You are now ready to add additional header information to the HTTP header and send the request to the server.

➢ **Send an HTTP Request**

1. Use the Developer Studio editor to open the file IC_2View.Cpp.
2. Add the following code to the body of the menu item handler **CIC_2View::OnInternetSendRequest**:

```
// Initialize our request header.
const TCHAR szHeaders[] =
    _T("Accept: text/*\r\nUser-Agent: IC_2\r\n");

BOOL rc;

// Send the request.
try {
    rc = m_pHttpFile->AddRequestHeaders(szHeaders);
    rc = m_pHttpFile->SendRequest();
}

catch (CInternetException * e)
{
    char buff[256];
    e->GetErrorMessage(buff,256);
    MessageBox(buff);
    e->Delete();
    return;
}

// Update the state of the menu items.
m_processState++;
```

You are now ready to receive, process, and display the data received from the HTTP server. To do so, implement these steps in the **OnInternetReadInformation** handler:

- Extract the header information from the server's message.
- Add to the document object.
- Extract the body of the message that is the targeted HTML file.
- Add the body contents to the document.
- Update the view to display the header and body contents.

➢ **Receive, process, and display the data received from the HTTP server**

1. Use the Developer Studio editor to open the file IC_2View.Cpp.
2. Add the following code to the body of the menu item handler **CIC_2View::OnInternetReadInformation**:

```
char s[1024];
CIC_2Doc* pDoc = GetDocument();

//
// Get the received header and prepend to view output.
//
DWORD bufsiz= 1024;
if (m_pHttpFile->QueryInfo(HTTP_QUERY_RAW_HEADERS_CRLF, s,
                            &bufsiz))
{
    char * p = strtok(s, "\n\r");
    // Extract line at a time.
    while (p)
    {
        pDoc->m_WebPage.AddTail(p);
        p = strtok(NULL, "\n\r");
    }
    // Delimiter between header and body.
    pDoc->m_WebPage.AddTail(
        "+=+=+=+=+=+ End of Header +=+=+=+=+=+=+");
}

//
// Read body.
//
m_pHttpFile->SetReadBufferSize(4096);

while(m_pHttpFile->ReadString(s, 1023))
{
    // Extract a line at a time.
    char * p = strtok(s, "\n\r");
    while (p)
    {
        pDoc->m_WebPage.AddTail(p);
        p = strtok(NULL, "\n\r");
    }
}

// Have view repaint itself.
pDoc->UpdateAllViews(NULL);

// Update state of the menu items.
m_processState++;
```

➢ **Build and test your application**

If you have an active connection to the Internet or an intranet, you should see the contents of the URL you requested displayed in the Web Browser control in your application's view class. The completed code for this exercise is in \Labs\C11\Lab02\Ex01.