原文はこちらをご覧下さい。

プロセス間通信の概要

クラス：wxServer, wxConnection, wxClient wxWidgets は、プロセス間通信とネットワークプログラミングに役立つ幾つかの異なるクラスを持っている。このセクションでは、ある１グループのクラス（DDE ライクなプロトコル）についてのみ説明するが、他にも、以下のような有用なクラスがある。

Classes: wxServer, wxConnection, wxClient wxWidgets has a number of different classes to help with interprocess communication and network programming. This section only discusses one family of classes -- the DDE-like protocol -- but here's a list of other useful classes:

- wxSocketEvent, wxSocketBase, wxSocketClient, wxSocketServer: 低レベルな TCP/IP API 用クラス
- wxProtocol, wxURL, wxFTP, wxHTTP: 一般的なインターネットプロトコル用クラス

* wxSocketEvent, wxSocketBase, wxSocketClient, wxSocketServer: classes for the low-level TCP/IP API.
* wxProtocol, wxURL, wxFTP, wxHTTP: classes for programming popular Internet protocols.

wxWidgets の DDE ライクなプロトコルは、Windows DDE をベースとした高レベルなプロトコルである。DDE ライクなプロトコルには、２つの実装がある。１つは Windows でのみ実行できる実際の DDE、もう１つは大部分のプラットフォームで実行可能な TCP/IP（ソケット）を使用したものである。それぞれのクラス名はともかくとし、API や実際の動作は同じであり、２つの実装を切り替えることが容易であることが分かるであろう。

wxWidgets' DDE-like protocol is a high-level protocol based on Windows DDE. There are two implementations of this DDE-like protocol: one using real DDE running on Windows only, and another using TCP/IP (sockets) that runs on most platforms. Since the API and virtually all of the behaviour is the same apart from the names of the classes, you should find it easy to switch between the two implementations.

<wx/ipc.h> をインクルードすることにより、IPC クラスの便利な同義が定義されることが分かる。DDE ベースかソケットベースの実装かによって、wxServer は wxDDEServer か wxTCPServer として定義される。wxClient や wxConnection も同様である。

Notice that by including <wx/ipc.h> you may define convenient synonyms for the IPC classes: wxServer for either wxDDEServer or wxTCPServer depending on whether DDE-based or socket-based implementation is used and the same thing for wxClient and wxConnection.

デフォルトでは、DDE 実装は Windows 環境下で使用される。DDE は１つのコンピュータでのみ動作する。異なるワークステーション間で IPC を使用したい場合、このヘッダをインクルードする前に wxUSE_DDE_FOR_IPC を 0 に定義する必要がある。これにより、Windows 環境下であっても、強制的に TCP/IP 実装を使用できる。

By default, the DDE implementation is used under Windows. DDE works within one computer only. If you want to use IPC between different workstations you should define wxUSE_DDE_FOR_IPC as 0 before including this header -- this will force using TCP/IP implementation even under Windows.

以下の説明では、wx... について説明するが、ほぼ同じことが wxTCP... や wxDDE... についても言える。

The following description refers to wx... but remember that the equivalent wxTCP... and wxDDE... classes can be used in much the same way.

３つのクラスが DDE ライクな API の中心である。

1. wxClient。クライアントアプリケーションを表し、クライアントプログラムでのみ使用される。
2. wxServer。サーバアプリケーションを表し、サーバプログラムでのみ使用される。
3. wxConnection。クライアントからサーバへのコネクションを表す。コネクションごとに、サーバとクライアントの両方でこのクラスのインスタンスを使用する。大部分の DDE トランザクションはこのオブジェクトを操作する。

Three classes are central to the DDE-like API:

+ wxClient. This represents the client application, and is used only within a client program.

+ wxServer. This represents the server application, and is used only within a server program.

+ wxConnection. This represents the connection from the client to the server - both the client and the server use an instance of this class, one per connection. Most DDE transactions operate on this object.

アプリケーション間のメッセージには、通常、３つの変数が使用される。コネクションオブジェクト、トピック名、アイテム名である。データ文字列は、幾つかのメッセージの４つ目の要素である。コネクションを確立するため（Windows 用語で、カンバセーション:会話)、クライアントアプリケーションはサーバオブジェクトにメッセージを送るため wxClient::MakeConnection を使用する。このとき、サーバを識別するための文字列型のサービス名、及び、コネクションが確立している間、トピックを識別するための文字列型のトピック名を使用する。Unix では、インターネットドメインソケットが通信に使用される場合には整数型のポート番号がサービス名として使用し、Unix ドメインソケットが作られる場合には固有のファイル名（存在してはならない。後で削除される）をサービス名として使用する。

Messages between applications are usually identified by three variables: connection object, topic name and item name. A data string is a fourth element of some messages. To create a connection (a conversation in Windows parlance), the client application uses wxClient::MakeConnection to send a message to the server object, with a string service name to identify the server and a topic name to identify the topic for the duration of the connection. Under Unix, the service name may be either an integer port identifier in which case an Internet domain socket will be used for the communications or a valid file name (which shouldn't exist and will be deleted afterwards) in which case a Unix domain socket is created.

セキュリティに関するメモ：インターネットドメインソケットを使用すると、これらのアクセス制御を完全に無くすことができないため、IPC を使用するには非常に危険である。できる限り Unix ドメインソケットを使用すべきである。

SECURITY NOTE: Using Internet domain sockets is extremely insecure for IPC as there is absolutely no access control for them, use Unix domain sockets whenever possible!

その時サーバは、コネクションを却下するか許可するか応答する。許可する場合には、サーバオブジェクトとクライアントオブジェクトは共に、コネクションが切断されるまで存在する wxConnection オブジェクトを作成する。そのコネクションオブジェクトは、クライアントーサーバ間のメッセージを送受信するために使用される。DDE メッセージを自分でハンドリングする場合には、wxConnection を派生し、仮想関数をオーバーライドする。

The server then responds and either vetoes the connection or allows it. If allowed, both the server and client objects create wxConnection objects which persist until the connection is closed. The connection object is then used for sending and receiving subsequent messages between client and server - overriding virtual functions in your class derived from wxConnection allows you to handle the DDE messages.

実際のサーバを作成するためにプログラマに必要とされること：

1. wxConnection の派生クラスを作成し、wxConnection のサーバ側に送られる様々なメッセージに対するハンドラ（例えば、OnExecute、OnRequest、OnPoke）を準備する。アプリケーションで実際に要求されるハンドルだけをオーバーライドすればよい。
2. wxServer の派生クラスを作成し、トピック引数に応じてコネクションを許可または拒否するよう、OnAcceptConnection をオーバーライドする。コネクションを許可する場合には、このメンバは派生されたコネクションクラスのインスタンスを作成し、返さなければならない。
3. 自身のサーバオブジェクトのインスタンスを作成し、サービス名を渡して Create を呼んで起動する。

To create a working server, the programmer must:

+ Derive a class from wxConnection, providing handlers for various messages sent to the server side of a wxConnection (e.g. OnExecute, OnRequest, OnPoke). Only the handlers actually required by the application need to be overridden.

+ Derive a class from wxServer, overriding OnAcceptConnection to accept or reject a connection on the basis of the topic argument. This member must create and return an instance of the derived connection class if the connection is accepted.

+ Create an instance of your server object and call Create to activate it, giving it a service name.

実際のクライアントを作成するためにプログラマに必要とされること：

1. wcConnection の派生クラスを作成し、wxConnection のクライアント側に送られる様々なメッセージに対するハンドラ（例えば、OnAdvise）を準備する。アプリケーションで実際に要求されるハンドルだけをオーバーライドすればよい。
2. wxClient の派生クラスを作成し、派生されたコネクションクラスのインスタンスを作成し、返すように、OnMakeConnection をオーバーライドする。
3. 自身のクライアントオブジェクトのインスタンスを作成する。
4. 使用する時には、ホスト名（Unix でのみ使用される。ローカルコンピュータの場合には、'localhost' を使用する）、サービス名、トピック名を wxClient::MakeConnection に渡して新しいコネクションを作成する。コネクションが成功すると、クライアントオブジェクトは、派生されたクラスのコネクションオブジェクトを作成するために OnMakeConnection を呼ぶ。
5. サーバにメッセージを送信するために、wxConnection のメンバ関数を使用する。

To create a working client, the programmer must:

+ Derive a class from wxConnection, providing handlers for various messages sent to the client side of a wxConnection (e.g. OnAdvise). Only the handlers actually required by the application need to be overridden.

+ Derive a class from wxClient, overriding OnMakeConnection to create and return an instance of the derived connection class.

+ Create an instance of your client object.

+ When appropriate, create a new connection using wxClient::MakeConnection, with arguments host name (processed in Unix only, use 'localhost' for local computer), service name, and topic name for this connection. The client object will call OnMakeConnection to create a connection object of the derived class if the connection is successful.

+ Use the wxConnection member functions to send messages to the server.

Data transfer

Examples

More DDE details

Data transfer

These are the ways that data can be transferred from one application to another. These are methods of wxConnection.

- Execute: the client calls the server with a data string representing a command to be executed. This succeeds or fails, depending on the server's willingness to answer. If the client wants to find the result of the Execute command other than success or failure, it has to explicitly call Request.
- Request: the client asks the server for a particular data string associated with a given item string. If the server is unwilling to reply, the return value is NULL. Otherwise, the return value is a string (actually a pointer to the connection buffer, so it should not be deallocated by the application).
- Poke: The client sends a data string associated with an item string directly to the server. This succeeds or fails.
- Advise: The client asks to be advised of any change in data associated with a particular item. If the server agrees, the server will send an OnAdvise message to the client along with the item and data.

The default data type is wxCF_TEXT (ASCII text), and the default data size is the length of the null-terminated string. Windows-specific data types could also be used on the PC.

Examples

See the sample programs server and client in the IPC samples directory. Run the server, then the client. This demonstrates using the Execute, Request, and Poke commands from the client, together with an Advise loop: selecting an item in the server list box causes that item to be highlighted in the client list box.

More DDE details

A wxClient object initiates the client part of a client-server DDE-like (Dynamic Data Exchange) conversation (available in both Windows and Unix).

To create a client which can communicate with a suitable server, you need to derive a class from wxConnection and another from wxClient. The custom wxConnection class will receive communications in a 'conversation' with a server. and the custom wxServer is required so that a user-overridden wxClient::OnMakeConnection member can return a wxConnection of the required class, when a connection is made.

For example:

```
class MyConnection: public wxConnection {
 public:
  MyConnection(void)::wxConnection() {}
   ~MyConnection(void) { }
   bool OnAdvise(const wxString& topic, const wxString& item, char *data, int size, wxIPCFormat
format)
  { wxMessageBox(topic, data); }
};

class MyClient: public wxClient {
 public:
  MyClient(void) {}
  wxConnectionBase *OnMakeConnection(void) { return new MyConnection; }
};
```

Here, MyConnection will respond to OnAdvise messages sent by the server by displaying a message box.

When the client application starts, it must create an instance of the derived wxClient. In the following, command line arguments are used to pass the host name (the name of the machine the server is running on) and the server name (identifying the server process). Calling wxClient::MakeConnection implicitly creates an instance of MyConnection if the request for a connection is accepted, and the client then requests an Advise loop from the server (an Advise loop is where the server calls the client when data has changed).

```
wxString server = "4242";
wxString hostName;
wxGetHostName(hostName);

// Create a new client
MyClient *client = new MyClient;
connection = (MyConnection *)client->MakeConnection(hostName, server, "IPC TEST");

if (!connection)
{
  wxMessageBox("Failed to make connection to server", "Client Demo Error");
  return NULL;
}
connection->StartAdvise("Item");
```