고급 COM 기술의 활용 (II)

(Using Advanced COM Techniques II.)

이번 장에서는 액티브 X 를 이용하여 여러가지 객체나 어플리케이션을 개발할 때 부딪힐 수 있는 문제점들과 이들에 대한 해결책을 제시할 것이다. 그렇게 방대한 내용은 아니지만, 소프트웨어를 개발하다 보면 실제로 아주 간단한 문제로 골머리를 싸맬 때가 매우 많다. 이번 장에서는 이런 문제들에 대해서 알아본다. 여기에서 소개하는 많은 팁들은 Inprise 의 뉴스 그룹에서 참고하였음을 미리 밝혀둔다.
참고로 델파이로 액티브 X 기술을 구현할 때의 여러 가지 팁과 FAQ 에 대한 정보는 Conrad Herrman 이 제공하는 홈 페이지에서 많이 얻을 수 있으므로 이를 참고하기 바란다. 이 홈페이지의 URL 은 Inprise 홈 페이지의 개발자 정보에서 찾을 수 있을 것이다.

Out of Process 논비주얼 COM 서버

여러가지 COM 서버를 작성하면서 지금까지 작성한 방법을 곰곰히 생각해보면, 결국에는 in-proc DLL 형태의 액티브 X 라이브러리로 만들거나 현재의 어플리케이션에 OLE 자동화 객체를 추가하는 형태로 만들었다. 이런 식으로 어플리케이션 프로젝트에 자동화 객체 위저드를 이용해서 추가한 경우에는 어떤 방식으로든 어플리케이션의 윈도우가 생성되고, 메시지 루프가 윈도우에 의해 사용된다.
예를 들어, out-of-process COM 서버를 만들면 언제나 좋든 싫든 폼이 하나 생성되어 버린다. 이럴 때에 물론 OnCreate 이벤트 핸들러에서 폼을 숨겨버리면 그만이지만, 불필요한 낭비임에는 틀림이 없다.
이런 경우에 윈도우가 생성되지 않고, in-proc 서버가 아닌 out-of-process 서버를 작성해서 사용하고 싶은 경우에는 어떻게 해야 할까 ?
이런 경우에는 Forms 유닛의 Application 객체를 변경해서 사용하되 비주얼 인터페이스를 사용하지 않게 하는 방법이 있다.
이를 위해서는 다음의 유닛을 사용하면 되는데, 이 유닛은 Inprise 의 뉴스 그룹에서 공개된 유닛인데, 작성자에 대한 정보를 기록해두지 않은 탓에 밝히지 못했음을 미리 말해둔다. 이 유닛으로 교체한 다음에 프로젝트 파일의 Application.Initialize 와 Application.Run 구문을 ServerApp.Initialize 와 ServerApp.Run 으로 대치하면 된다. 또한, 어플리케이션에서 Application.ProcessMessage 루틴이 사용되는 부분이 있다면 ServerApp 에 대한 내용으로 변경해야 하는 것은 물론이다.

```
unit ServApp;
```

```
interface

type
  TServerApp = class(TObject )
  protected
    procedure DoTerminate(var Shutdown: Boolean);
  public
    function ProcessMessages: Boolean;
    procedure Initialize;
    procedure Run;
  end;

var
  ServerApp: TServerApp;

implementation

uses
  ComServ;

function TServerApp.ProcessMessages: Boolean;
var
  Msg: TMsg;
begin
  Result := true;
  while PeekMessage(Msg, 0, 0, 0, PM_REMOVE) do
    if   (Msg.Message = WM_QUIT) then
      Result := false
    else
    begin
      TranslateMessage(Msg);
      DispatchMessage(Msg);
    end;
end;
```

```
procedure TServerApp.DoTerminate(var Shutdown: Boolean);
begin
  if Shutdown and not CallTerminateProcs then
    Shutdown := False;
end;
```

CallTerminateProcs 루틴은 델파이의 COM 시스템을 종료하는 역할을 한다.

```
procedure TServerApp.Initialize;
begin
  if (ComServer.StartMode = smStandalone) then Halt;
  if (InitProc <> nil) then
    TProcedure(InitProc);
  ComServer.OnLastRelease := ServerApp.DoTerminate;
end;
```

InitProc 은 COM 서버 시스템을 초기화하기 위해서 필요하다. 즉, 이 부분이 핵심이라고 할 수 있는데, COM 서버가 stand-alone 으로 시작하지 못하도록 하고 마지막 OnLastRelease 이벤트에 DoTerminate 메소드를 실행하도록 대입하여 COM 서버가 종료할 때 실행되도록 한다.

```
procedure TServerApp.Run;
begin
  while ProcessMessages do;
end;
```

Run 메소드는 간단히 ProcessMessage 루틴을 반복하도록 하면 폼이 필요 없이 out-of-process COM 서버를 쉽게 구현할 수 있다.

```
initialization
  ServerApp := TServerApp.Create;

finalization
  ServerApp.Free;
end.
```

구현한 내용을 보면 무척 간단하다는 것을 알 수 있을 것이다. 별거 아닌 것 같지만, 필요에 의해서 이런 유닛을 찾아 다녀 보면 막상 쓸만한 것이 없는 경우가 많고, 그렇다고 직접 작성하려하면 막막한 경우가 많다.
이 주제 역시 해결 방법을 보면 별 것 아니지만, 막상 만들어 보려고 하면 쉽지 않은 것이다. 의외로 많이 사용할 가능성이 있는 해결 방법이므로 꼭 기억했다가 사용하기 바란다.

가변형 변수와 스트림간 통신

액티브 X 기술을 이용하여 클라이언트/서버 어플리케이션을 제작하다 보면, 간혹 스트림을 통해 바이너리 파일을 전송할 필요가 있을 때가 있다. 이럴 때에는 DCOM 에서 지원하는 데이터 형에 TStream 클래스가 호환되지 않기 때문에 가변형 변수를 사용하여 전송을 해야 한다.
즉, 다시 말하면 서버 측에서는 파일을 메모리 스트림에 읽어오고, 이를 가변형 변수로 형변환을 한 뒤에 클라이언트로 전송하고, 클라이언트에서는 이렇게 넘어온 가변형 변수를 메모리 스트림으로 변경하면 된다.
다음에 스트림과 가변형간의 형 변환을 담당하는 루틴을 소개한다.

```
function StreamToVariant(Stream:TMemoryStream): Variant;
var
  Data: Pointer;
begin
  Result := VarArrayCreate([0, Stream.Size - 1], varByte);
  Data := VarArrayLock(Result);
  try
    Move(Stream.Memory^, Data^, Stream.Size);
  finally
    VarArrayUnlock(Result);
  end;
end;

function VariantToStream(V: Variant): TMemoryStream;
var
  Data: Pointer;
begin
  Result := TMemoryStream.Create;
  Data := VarArrayLock(V);
```

```
  try
    Result.WriteBuffer(Data^, VarArrayHighBound(V, 1) + 1);
  finally
    VarArrayUnlock(V);
  end;
  Result.Seek(0, soFromBeginning);
end;
```

IOleClientSite 인터페이스의 활용

IOleClientSite 인터페이스를 이용하면 컨테이너에 임베드된 객체의 정보를 얻을 수 있다. 이 인터페이스를 얻기 위한 함수를 다음과 같이 구현할 수 있다.

```
function ClientSite(obj: IUnknown): IOleClientSite;
var
  Site: IOleClientSite;
  OleObj: IOleObject;
begin
  if (obj.QueryInterface(IOleObject, OleObj) = S_OK) and
     (OleObj.GetClientSite(Site) = S_OK) then
    Result := Site
  else
    Result := nil;
end;
```

obj 파라미터는 액티브 X 컨트롤을 지정한다. 예를 들어, 다음과 같이 사용할 수 있다.

```
type
  TButtonX = class(TActiveXControl)
  ...
  end;

... (중략)

procedure TButtonX.Click;
var
```

```
  Site: IOleClientSite;
begin
  Site := ClientSite(Self);
end;
```

액티브 X 컨트롤을 개발한 뒤에 이를 실제로 사용하게 되면, 실행되는 모드가 런타임인지 아니면 디자인 타임인지를 구별할 필요가 있을 때가 있다. 이를 파악하기 위해서는 컨트롤의 컨테이너의 UserMode 앰비언트 프로퍼티를 이용해야 한다.
이때에도 앞의 ClientSite 함수를 이용하면 컨테이너의 모드를 쉽게 알 수 있다. 다음 함수는 컨테이너가 디자인 모드이면 True 를 반환한다.

```
function IsControlInDesignMode(obj: IUnknown): Boolean;
var
  Mode: Boolean;
begin
  try
    Mode := not ((ClientSite(obj) as IAmbientDispatch).UserMode);
  except
    Mode := False;
  end;
  Result := Mode;
end;
```

Safe for scripting/safe for initializing 의 지원

‘safe for scripting’과 ‘safe for initializing’이란 코드 사인을 통한 기본적인 보안과 함께 사용될 수 있는 Object Safety 라는 2 차 보안을 나타내는 용어이다.
컨트롤을 HTML 페이지에서 다운로드하면 페이지가 자바 스크립트나 VB 스크립트 등을 이용할 수도 있고, 컨트롤에 대한 프로퍼티 값들이 포함된다. 이때 제작한 컨트롤의 일부 메소드나 프로퍼티를 잘못 사용할 때 클라이언트 컴퓨터에 나쁜 영향을 미칠 수 있다면 이러한 메소드나 프로퍼티가 잘못 사용되지 않도록 해야 할 것이다.
컨트롤이 ‘safe for scripting’으로 표시된다는 의미는 객체가 OLE 자동화를 통해 안전하게 자동화될 수 있다는 의미이다. 즉, 객체를 자동화하는 각종 스크립트에 대해서 클라이언트 컴퓨터에 나쁜 영향을 주지 않는다는 의미이다. 그리고, ‘safe for initializing’으로 표시된다는 의미는 객체의 프로퍼티나 데이터가 어떤 지속성 저장소(persistent storage)에서도 저장되었다가 복원될 수 있다는 의미이다. 이들 자체에 대한 보다 자세한 사항은 마이크로소프

트에서 제공하는 자료 들을 참고하기 바란다.
그렇다면 컨트롤에 ‘safe for scripting/initializing’을 표시하는 방법에 대해서 알아보도록 하자. 여기에는 크게 2 가지 방법이 존재하는데 첫번째 방법은 컨트롤이 언제나 스크립팅과 초기화에 안전하다고 표시하는 것이고, 다른 하나는 컨트롤이 safe mode 와 unsafe 모드를 전환할 수 있도록 표시하는 방법이다.

1. 컨트롤에 대한 적절한 레지스트리 키를 설치한다.

컨트롤이 설치될 때 레지스트리를 변경하는 방법에 대해서는 클래스 팩토리를 오버라이드하여 구현한다는 것을 이미 앞에서 설명한 바 있다. 그러므로, 해당되는 액티브 X 컨트롤의 클래스 팩토리 클래스를 오버라이드하여 ‘safe for scripting/safe for initialization’을 표시하도록 구현하면 된다.
다음의 유닛에서 이들 클래스 팩토리를 상속하여 구현하였다. 이 유닛은 Conrad Herrman 이 DAX FAQ 를 통해서 공개한 유닛임을 미리 밝혀 두며, 소스 코드에 대한 설명은 이미 29 장에서 UpdateRegistry 메소드를 오버라이드하여 구현하는 방법에 대해 설명한 바 있기 때문에 생략하겠다.

```
unit SafeFactory;
interface
uses ComObj, ActiveX, AXCtrls;

const
  CATID_SafeForScripting: TGUID = '{7DD95801-9882-11CF-9FA9-00AA006C42C4}';
  CATID_SafeForInitializing: TGUID = '{7DD95802-9882-11CF-9FA9-00AA006C42C4}';

type
  TSafeActiveFormFactory = class(TActiveFormFactory)
    procedure UpdateRegistry(Register: Boolean); override;
  end;

  TSafeActiveXControlFactory = class(TActiveXControlFactory)
    procedure UpdateRegistry(Register: Boolean); override;
  end;

implementation
```

```
procedure AddSafetyKeys(const ClassID: TGUID);
var
  ClassKey: string;
begin
  ClassKey := 'CLSID₩' + GUIDToString(ClassID);
  CreateRegKey(ClassKey + '₩Implemented Categories', '', '');
  CreateRegKey(ClassKey + '₩Implemented Categories₩' + GUIDToString(
      CATID_SafeForScripting), '', '');
  CreateRegKey(ClassKey + '₩Implemented Categories₩' + GUIDToString(
      CATID_SafeForInitializing), '', '');
end;

procedure RemoveSafetyKeys(const ClassID: TGUID);
var
  ClassKey: string;
begin
  ClassKey := 'CLSID₩' + GUIDToString(ClassID);
  DeleteRegKey(ClassKey + '₩Implemented Categories₩' + GUIDToString(
      CATID_SafeForInitializing));
  DeleteRegKey(ClassKey + '₩Implemented Categories₩' + GUIDToString(
      CATID_SafeForScripting));
  DeleteRegKey(ClassKey + '₩Implemented Categories');
end;

{TSafeActiveFormFactory}
procedure TSafeActiveFormFactory.UpdateRegistry(Register: Boolean);
begin
  if Register then
  begin
    AddSafetyKeys(ClassID);
    inherited UpdateRegistry(Register);
  end
  else
  begin
    RemoveSafetyKeys(ClassID);
    inherited UpdateRegistry(Register);
```

```
  end;
end;

{TSafeActiveXControlFactory}
procedure TSafeActiveXControlFactory.UpdateRegistry(Register: Boolean);
begin
  if Register then
  begin
    AddSafetyKeys(ClassID);
    inherited UpdateRegistry(Register);
  end
  else
  begin
    RemoveSafetyKeys(ClassID);
    inherited UpdateRegistry(Register);
  end;
end;

end.
```

이 클래스 팩토리를 이용하기 위해서는 액티브 X 라이브러리의 유닛의 uses 절에 SafeFactory.pas 유닛을 추가하고, initialization 섹션을 다음과 같이 클래스의 이름만 변경해주면 된다.

```
initialization
  TSafeActiveXControlFactory.Create(ComServer, TSampleActiveX, TSampleControl,
    Class_SampleActiveX, 1, '', 0, tmApartment);
end.
```

2. 컨트롤에서 IObjectSafety 인터페이스를 구현하는 방법

IObjectSafety 인터페이스를 구현하면 컨트롤이 safety 를 요구하지 않는 컨테이너에서 동작할 때에는 unsafe 메소드나 프로퍼티를 사용할 수 있게 할 수 있다. 그렇지만, IE 와 같이 보안을 요구하는 컨테이너에서는 IObjectSafety 인터페이스를 이용하여 컨트롤이 safe mode 로 전환하도록 요구하고, 모든 unsafe 메소드와 프로퍼티를 사용 불가능하게 만들 수 있다.

델파이 폼을 액티브 폼으로 전환하기

델파이 폼을 액티브 폼으로 변경하는 요령에 대해서 설명하고자 한다. 많은 경우에 있어서 처음부터 액티브 폼으로 개발하는 경우도 있겠지만, 이미 완성된 어플리케이션을 액티브 폼으로 변경하는 작업을 하고자 하는 경우도 많을 것이다. 이럴 때에는 다음과 같은 방법을 이용하여 액티브 폼으로 변경할 수 있다.

- 폼에 있는 모든 컴포넌트를 선택하고 이를 복사했다가 액티브 폼에 붙여넣는 방법

이 방법은 컴포넌트를 폼에 옮기는데에는 큰 문제가 없지만, 컨트롤과 연결된 코드는 복사되지 않기 때문에 이들을 적절하게 설정하는 것이 가장 중요한 작업이 된다.

- 컴포넌트 템플릿을 활용하는 방법

폼에 있는 모든 컴포넌트를 메뉴를 선택하고 Component|Create Component Template 메뉴를 선택하여 템플릿을 생성하면 VCL 컴포넌트 팔레트에 Template 탭이 생성되면서 컴포넌트로 등록될 것이다. 액티브 폼을 열고 이 컴포넌트를 폼에 떨어뜨리면 된다.
이 방법의 장점은 컨트롤과 함께 연결된 코드가 같이 복사된다는 점이다.

- 직접 TForm 클래스를 액티브 폼으로 변경시키는 방법

앞서 설명한 방법과 같이 컨트롤을 복사하지 않고, 코드를 수정함으로써 액티브 폼으로 변경하는 방법이다. 이 방법의 장점은 표준 델파이 어플리케이션으로 일단 개발과 테스트를 마치고 간단히 액티브 폼 버전으로 변경하여 사용할 수 있다는 점이다.
변경 원칙은 다음과 같다. 이 방법은 Conrad Herrman 이 공개한 것으로 꽤 유용하게 적용할 수 있다.

1. 표준 폼 어플리케이션을 디렉토리에 저장한다.
2. 액티브 폼을 저장할 서브 디렉토리를 생성하고, 새로운 액티브 폼 프로젝트를 생성하여 저장한다.
3. 액티브 폼 프로젝트에서 프로젝트 관리자를 선택하고 표준 폼 어플리케이션 프로젝트가 저장된 디렉토리에서 모든 폼과 데이터 모듈을 프로젝트에 추가하도록 한다.
4. 액티브 폼의 OnCreate 이벤트 핸들러를 다음과 같이 작성한다.

```
procedure TActiveForm1.FormCreate(Sender: TObject);
```

```
begin
        //이 코드는 정상적인 델파이 TForm 인 자식 폼을 생성한다.
  ChildForm := TForm1.Create(Self);
  ChildForm.Parent := Self;
  ChildForm.Align := alClient;
  ChildForm.BorderStyle := bsNone;
  ChildForm.Visible := True;
end;
```

5. uses 절에서 폼의 유닛 파일을 추가하고, 자식 폼을 public 섹션에 추가한다.

```
type
  TActiveForm1 = class ...
  ....
  public
    ChildForm: TForm1;
  ....
  end;
```

폼을 컴파일하고 테스트한다.

액티브 폼과 IE 4.0

델파이의 액티브 폼은 일종의 액티브 X 컨트롤이지만, 의외로 IE 4.0 과 충돌하는 경우가 많다. 이 문제는 델파이의 액티브 폼 위저드에도 약간의 문제가 있지만 IE 4.0 자체가 표준과는 벗어난 여러가지 문제점을 많이 안고 있기 때문이기도 하다. 그러므로, 이 문제를 완전히 해결할 수는 없지만 몇 가지 고려할 점들에 대해서 논의하고자 한다.
먼저 델파이 3 에서 문제가 되었던 것 중에서 델파이 4 에서 해결된 것들에 대해서 알아보고, 계속해서 고려해야 할 점을 알아보도록 하자.

1. 쓰레딩 모델

델파이 3 는 기본적으로 단일 쓰레드 모델 만을 지원했기 때문에, IE 4.0 의 여러 인스턴스를 메모리에 띄울 때 충돌이 일어나는 경우이다. 이 문제는 델파이 4 에서 쓰레딩 모델로 apartment 쓰레딩 모델을 선택할 수 있게 되면서 많이 해결되었다. 보통 디폴트로 이 모드를 사용하게 되지만, IE 4.0 과의 충돌이 있다면 먼저 쓰레딩 모델을 검토하기 바란다.

2. 액티브 폼 내부 컨트롤의 포커스 설정 문제

여러 개의 컨트롤을 가지고 있는 액티브 폼에서 하나의 컨트롤을 클릭하여 포커스를 주고, 다른 어플리케이션을 활성화 시켰다가 다시 IE 4.0 을 활성화하면 이전에 포커스를 가졌던 컨트롤이 포커스를 잃게 된다.
이 문제는 액티브 폼 자체가 UI-active 컨트롤이기 때문이다. 그러므로, 프레임이 활성화 되면 컨트롤이 IInPlaceActiveObject.OnFrameWindowActivate 메소드가 호출되는데 델파이 3 의 위저드에서는 이 메소드가 단지 InPlaceActivate(True) 메소드만을 호출하기 때문에, 컨트롤이 이미 UI-active 한 경우에는 아무런 영향을 미치지 못한다. 그러므로 앞서 설명한 문제를 해결하기 위해서는 폼의 활성화된 자식 컨트롤에 포커스를 설정해야 하는데 델파이 4 에서 이 점이 수정되었다.

3. 탭/백스페이스 키가 제대로 동작하지 않을 때

IE 4.0 에서 액티브 폼을 띄운 뒤에 탭/백스페이스 키를 이용할 때 제대로 동작하지 않는 문제가 발생하는 경우가 있다. 이 문제를 해결하기 위해서는 액티브 폼의 타입 라이브러리를 동작시키고 TabsOn 이라는 새로운 메소드를 인터페이스에 추가한다. 이 메소드의 선언은 'procedure TabsOn' 으로 특별한 파라미터나 반환값은 지정할 필요가 없다. 그리고, 이 메소드에 대한 ID 를 부여한다. 그리고 나서 액티브 폼을 구현한 유닛의 protected 섹션에 선언되어 있는 TabsOn 선언부를 다음과 같이 public 섹션으로 옮긴다.

```
  public
    procedure TabsOn; safecall;
```

그리고, TabsOn 프로시저의 구현 부분을 다음과 같이 수정한다.

```
procedure TActiveFormX.TabsOn;
begin
  (ComObject As IOleInPlaceActiveObject).OnFrameWindowActivate(True);
end;
```

그리고, 다소 귀찮기는 하지만 폼에 있는 모든 컨트롤의 OnClick 이벤트에 TabsOn 을 호출하도록 추가하여 액티브 폼을 클릭할 때마다 UI 를 활성화 시키도록 한다.
다른 해결 방법으로는 Conrad Herrman 이 제시한 방법이 있는데, 그의 해결책은 폼에 WM_MOUSEACTIVATE 메시지에 대한 핸들러를 설치해서, 이 메시지가 발생할 때마다 UI

를 활성화하는 것이다. 이렇게 하면, 핸들러가 설치된 자식 컨트롤 들이 클릭될 때마다 활성화되어 제대로 포커스를 가지게 된다.
메시지 핸들러는 다음과 같이 구현한다. 먼저 다음과 같이 핸들러를 선언하고 이를 구현하면 된다.

```
procedure WMMouseActivate(var msg: TWMMouseActivate); message WM_MOUSEACTIVATE;

... (중략)

procedure TActiveFormX.WMMouseActivate(var msg: TWMMouseActivate);
begin
  inherited;
  if (msg.Result = MA_ACTIVATE) or (msg.Result = MA_ACTIVATEANDEAT) then
  begin
    DoUIActivate;
  end;
end;

procedure TActiveFormX.DoUIActivate;
begin
  if (ComObject <> nil) then
  begin
    (ComObject as IOleObject).DoVerb( OLEIVERB_UIACTIVATE,
      nil, nil, 0, 0, PRect(nil)^);
  end;
end;
```

이 방법은 대부분의 경우에 적용되지만 RichEdit 컨트롤에서는 사용할 수 없다. 이 경우에는 OnClick 메소드를 이용한 방법을 사용해야 한다. 액티브 폼에서 탭 키를 누르면 실제로 컨트롤에서 이동은 일어나지만, 이를 실제로는 제대로 보여주지 못하기 때문이다. 또한, UI가 활성화되어도 자신 컨트롤에 포커스를 제대로 설정하지 못하기 때문에 이런 현상이 나타나는 것이므로 앞서와 같은 방법으로 이 문제를 해결해야 한다.

액티브 X 서버 배포에 관한 문제

몇 가지 인터페이스를 구현한 액티브 X 서버를 배포할 때에는 배포할 때 염두에 두어야 할

몇 가지 사항이 있다. IStrings, IProvider, IDataBroker 인터페이스를 사용하거나 델파이 폰트, 색상, 문자열, 그림 프로퍼티 페이지를 사용한 경우에는 볼랜드의 표준 VCL 타입 라이브러리를 같이 배포해야 한다.
이 라이브러리는 STDVCL32.DLL 라이브러리와 STDVCL32.TLB 라고 하는 타입 라이브러리로 존재한다. 이들은 모두 윈도우의 시스템 디렉토리에 위치하며, 모두 시스템 레지스트리에 등록되어야 한다.

액티브 X 컨트롤의 코드 다운로드 문제

29 장에서도 설명했듯이 액티브 X 컨트롤을 배포할 때 Web Deployment options 대화 상자에서 적절한 정보를 설정하고, Web Deploy 명령을 선택하면 되는데, 막상 홈 페이지에 올려 놓고 이 홈 페이지를 클라이언트에 가서 브라우저를 띄우고 접근하면 커다란 붉은 X 자만 볼 수 있는 경우가 많다.
제작한 액티브 X 컨트롤을 웹 브라우저에 띄우기 위해서 Web Deploy 명령을 선택하면 HTML 문장에 <OBJECT> 태그가 추가되면서 다음과 같이 액티브 X 컨트롤에 대한 정보가 추가된다.

```
<OBJECT
        classid="clsid:29D37F03-F02F-11D0-ACB2-0080C7316F20"
        codebase="http://www.SampleSite.com/MyControl.ocx#version=1,0,1,0"
        width=350
        height=250
        align=center
        hspace=0
        vspace=0
>
</OBJECT>
```

이 태그에서 #version 부분은 옵션인데, 이 내용이 빠지면 버전에 상관이 없다는 의미이다. 실제 액티브 X 컨트롤을 지칭하는 것이 ClassID 이다. 그러므로, IE 가 지정된 ClassID 의 컨트롤이 클라이언트에 설치되어 있으면, 지정된 버전과 일치하는 것인지 알아보고 로컬 버전이 일치하면 컨트롤을 생성하기 위해 로컬 복사본을 이용하게 된다.
만약 HTML 페이지에서 요구하는 버전이 현재 설치된 것보다 새 버전이거나 ClassID 에 지정된 컨트롤이 아직 설치되지 않은 경우에는 IE 가 컨트롤을 생성하기 전에 컨트롤을 다운로드하고 이를 설치하게 된다.
이때 코드 베이스는 로컬 기계의 Windows/OCCache 디렉토리나 서브 디렉토리에 복사하

는데 이때 URI 가 HTTP(디폴트)를 이용하거나, FTP, FILE URI 등을 이용할 수 있다. 여기서 원격 서버를 찾을 수 없거나, 로컬 디스크의 디스크 공간이 보족한 경우, 그리고 codebase 의 이름이 잘못 설정된 경우(서버가 대소문자를 가리는 경우에 흔함)에 이 과정이 실패할 수 있다. .

액티브 X 컨트롤을 다운로드하기 전에 IE 가 코드 signature 를 확인하게 된다. 여기에서 꽤 많은 브라우저의 설정이 잘못된 경우에 액티브 X 컨트롤을 볼 수 없다. 보안 레벨을 High 로 설정한 경우에는 IE 가 컨트롤이 코드 사인되지 않았으면 설치를 하지 않는다. 보안 레벨이 Medium 으로 설정되면 설치할 것인지 물어보는 대화상자를 보여 주고, 컨트롤을 다운로드하게 된다. 보안 레벨이 Low 로 설정되면 코드 사인되지 않은 컨트롤도 바로 설치된다.

이런 과정을 거쳐 다운로드된 codebase 는 Windows/OCCache 디렉토리에 설치된다. 설치되는 codebase 는 Object 택그의 CODEBASE 태그에 의해 그 종류가 결정된다. 만약 파일이 DLL 인 경우(보통 확장자 .OCX) 파일 하나를 가리키며, 파일이 .CAB 인 경우에는 여러 파일이 포함되어 있다. 그리고, codebase 가 .INF 파일인 경우에는 codebase 를 구성하는 파일의 리스트를 설명한다.

DLL 인 경우에는 설치 과정에서 DLL 파일을 로드한 뒤에 DllRegisterServer 함수를 호출하여 컨트롤을 생성하게 된다. 이 과정에서 DLL 파일이 LoadLibrary 함수에 의해 로드되지 않을 수가 있는데 이것은 DLL 이 라이브러리 DLL 이나 델파이 패키지, 시스템 DLL 과 같은 다른 파일에 의존적인데 이 파일들에 접근할 수 없는 경우이다. DLL 이 로드되지만 초기화에 실패하는 경우도 있는데, 이것은 환경 설정에도 문제가 있을 수 있지만 유닛의 초기화 섹션에서 예외가 발생하기 때문일 수도 있다.

또다른 문제로는 컨트롤이 제대로 등록되지 않는 경우에 발생할 수 있다. DLL 이 등록되지 않는 원인으로 가장 흔한 것은 NT 클라이언트를 사용할 경우에 사용자 권한이 레지스트리를 변경할 수 있는 권한이 없는 경우이다.

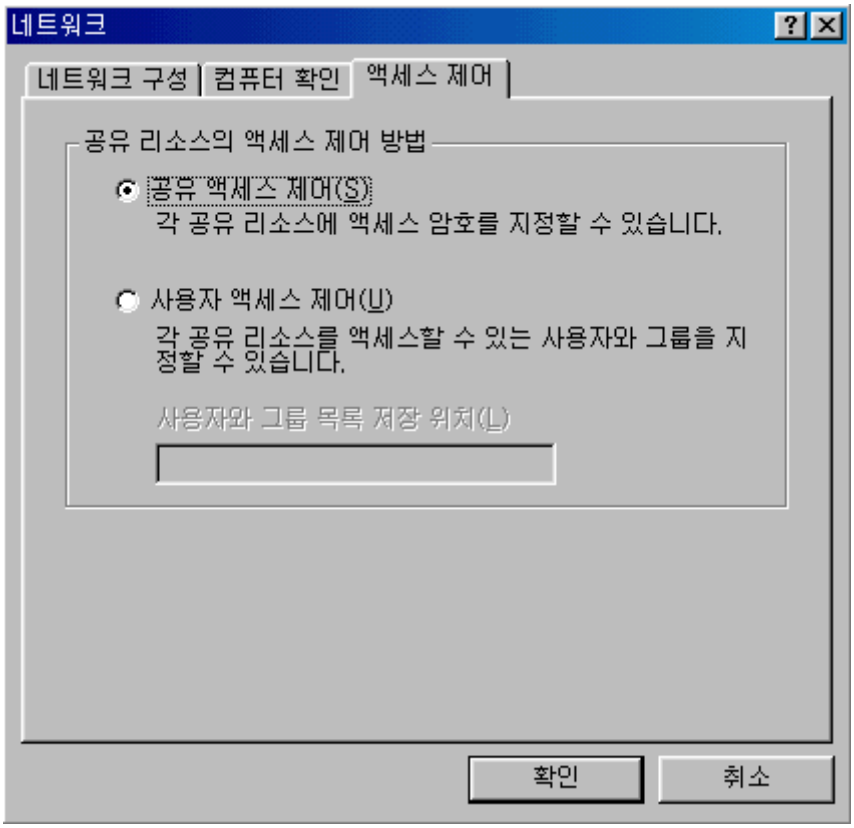
컨트롤이 이런 과정을 거쳐서 설치되고 생성되면 IE 는 컨트롤의 인스턴스를 생성하려고 시도한다. 이 과정에서 문제가 발생하는 이유로는 HTML 파일에서 지정된 ClassID 가 잘못 지정된 경우와 메모리나 리소스가 부족한 것이 원인일 수 있다.

DCOM 환경 설정

COM 객체와 액티브 X 컨트롤과 액티브 폼 객체를 훌륭하게 생성하고, 이를 이용하여 인트라넷이나 인터넷을 통해서 사용하려고 할 때 문제가 되는 이유 중에서 가장 커다란 비중을 차지하는 것은 뜻 밖에도 DCOM 의 환경 설정을 제대로 하지 못했기 때문인 경우가 많다. 그러면, DCOM 클라이언트/서버의 환경 설정에 대해서 중요한 사항을 알아보도록 하자.

- 서버의 환경 설정

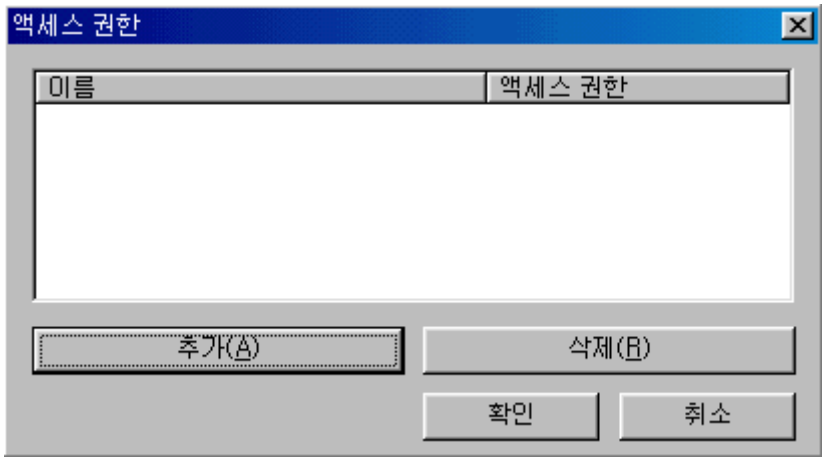
DCOM 의 서버는 사용자 레벨(user level)과 공유 레벨(share level) 접근이 가능하도록 환경설정을 할 수 있다. 이 설정을 변경하기 위해서는 제어판의 네트워크 애플릿을 실행하고, 다음과 같이 엑세스 제어 페이지에서 설정을 변경할 수 있다.



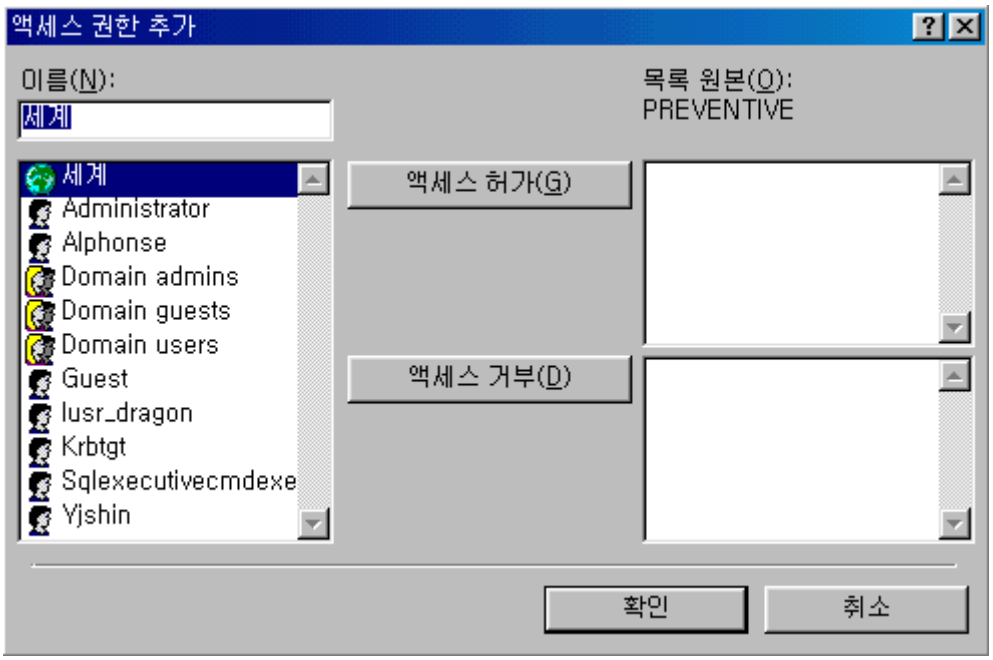
만약 윈도우 NT 서버를 이용한 사용자 인증이 가능하다면, 사용자 엑세스 제어를 선택하고, 적절한 NT 서버를 지정하여 사용자와 그룹을 이용하도록 하면 된다. NT 서버와 관계 없이 DCOM 을 이용하기 위해서는 공유 엑세스 제어를 선택하도록 한다.

윈도우 95 에서 DCOM 을 사용하기 위해서는 반드시 DCOM for Win95 를 설치해야 하는데 이 설치 파일은 http://www.microsoft.com/com/dcom95/download.htm 에서 구할 수 있다. 그렇지만 필자의 생각으로 DCOM 을 사용하기 위해서는 윈도우 NT 4.0 SP 3, 윈도우 98 이상에서 사용하기를 권하고 싶다.

사용자 엑세스 제어를 선택한 경우에는 DcomCnfg.exe 유틸리티를 실행하고, 이 유틸리티의 기본 보안(Default Security) 페이지를 선택하고, 기본값 편집(Edit Default) 버튼을 클릭하면 다음과 같이 액세스 권한(Access Permissions) 대화 상자가 나타날 것이다.



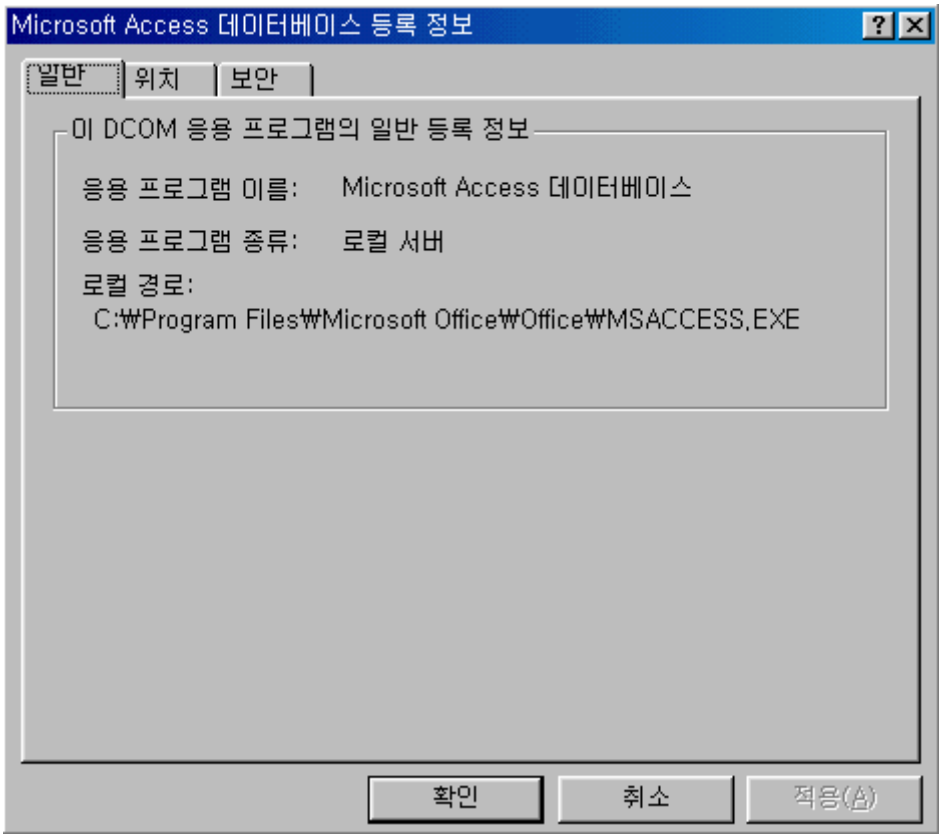
이 대화 상자에서 '추가' 버튼을 클릭하면 다음과 같이 서버에서 관리하고 있는 모든 사용자 목록이 나타나는데, 서버 어플리케이션에 접근해야 하는 모든 사용자 들에게 접근을 허용하도록 설정한다.



여기에서 '세계'를 선택하면 모든 사용자에게 모두 사용 가능하도록 설정하는 것이다. 윈도우 NT 의 경우에는 다소 화면이 다르게 나타날 수 있는데 '세계'에 해당되는 것은 'Everyone'이다. 참고로 필자는 윈도우 NT 5.0 영문 베타를 사용하고 있는 관계로 한글 NT 의 해당 이름을 알지 못하므로 한글 NT 를 사용하고 있는 독자들은 한글로 해당되는 이름을 찾아보기 바란다.

이런 방법을 사용하지 않고, 서버 어플리케이션을 설치하고 각 어플리케이션마다 접근 허용을 다르게 설정하는 방법도 있다.

이럴 경우에는 DCOM 구성 등록 정보 윈도우의 응용 프로그램 탭에서 어플리케이션을 선택하고 '등록정보' 버튼을 클릭하면 다음과 같이 이 어플리케이션에 대한 정보를 변경할 수 있는 대화상자가 나타날 것이다.



여기에서 '보안' 탭을 선택하면 앞에서 전체적으로 엑세스 권한을 설정했던 것과 동일한 대화상자 들이 나타날 것이다. 이를 이용하여 각 어플리케이션 별로 접근 레벨을 다르게 설정할 수 있다.

그리고, 이 예에서는 나타나지 않았지만 '응용프로그램 실행' 탭이 있는 경우에는 이 탭에서 선택할 수 있는 체크 박스가 '데이터가 있는 컴퓨터', '현재 컴퓨터', '컴퓨터 선택'의 3 가지가 나타나는데, 여기서 '현재 컴퓨터' 박스는 선택하면 안된다. 이를 선택하면 현재 컴퓨터에 있는 서버만 실행이 가능하므로 DCOM 으로 설정한 의미가 없다. 그리고, '현재 컴퓨터'와 '컴퓨터 선택'이 모두 선택된 경우에는 현재 컴퓨터가 우선한다. 그러므로, DCOM 서버로 작성한 서버 프로그램이 서버에서 원격으로 실행되게 하려면 '컴퓨터 선택' 탭을 선택하고 서버의 DNS 주소, IP 주소, 호스트 이름 등을 지정하면 된다.

참고로 윈도우 NT 에서 DComConfg.exe 유틸리티를 사용할 경우에는 Identity, Endpoints, HTTP 탭이 더 있는데 이 중에서 Identity 탭을 선택하고 'The interactive user'로 설정하는 것이 좋다.

기본적인 설정이 끝났으면 NT 에서 사용자 관리자(User Manager)를 실행하여, Guest 계정을 선택하고 이 계정의 정보에서 'Accound Disabled' 체크 박스가 선택되어 있으면 이를 제거해야 한다.
공유 엑세스 제어를 선택한 경우에는 레지스트리 값을 변경할 필요가 있다.
HKLM₩Software₩Microsoft₩OLE 키의 내용 중에서 EnableRemoteConnect = "Y", LegacyAuthenticationLevel = 1 로 설정하도록 한다. 이 작업은 사용자 엑세스 제어를 사용할 때에는 DcomCnfg.exe 유틸리티를 통해서 수정이 가능하기 때문에 매뉴얼로 키를 변경할 필요가 없다. DcomCnfg.exe 유틸리티에서는 기본 등록 정보 탭에서 인증 수준을 '없음'으로 선택하고, 기본 보안 탭에서 '원격 연결 사용' 체크 박스를 선택하면 된다 (디폴트로 선택되어 있다).
그리고, 델파이를 이용하여 작성한 어플리케이션 중에서 MIDAS 와 같이 필요한 파일이 더 있는 경우에는 해당 파일을 System 또는 System32 디렉토리에 복사하도록 한다. 대표적인 파일로는 DBCLIENT.DLL, STDVCL32.DLL 파일 등이 있다.
이제 서버 어플리케이션을 적당한 로컬 드라이브에 복사하고, 필요에 따라서 BDE 나 SQL Link 등을 설치하고 필요한 앨리어스 등을 생성한다.
서버 어플리케이션을 한번 실행하면 레지스트리에 등록되므로 쉽게 사용할 수 있고, in-process 서버는 DCOM 에서 사용되지 않는다.
만약 DCOM 95 1.0 버전을 이용하고 있다면, RPCSS.EXE 의 단축 아이콘을 시작 폴더에 위치시키는 것이 좋다.
이런 환경 설정의 변경이 효력을 발생하려면 시스템의 재시작이 필수적이다.
이제 서버 어플리케이션을 실행하면 클라이언트와의 연결이 가능하다.

- 클라이언트의 환경 설정

서버에 비해 클라이언트의 환경 설정은 더 간단하다. 먼저, 윈도우 95 를 사용하는 독자라면 앞서 소개한 사이트에서 DCOM95 를 다운로드 받아 설치하기 바란다.
기본적인 설정 방법은 서버와 동일하므로 앞의 내용을 참고하기 바란다.
공유 방법으로 공유 엑세스 제어를 사용하는 클라이언트인 경우에는 서버에서와 마찬가지로 레지스트리의 내용을 편집할 필요가 있다. HKLM₩Software₩Microsoft₩OLE 키에서 LegacyAuthenticationLevel = 1 로 설정한다.
설정이 끝났으면, 클라이언트 어플리케이션을 설치하고 실행하면 된다. 델파이로 작성한 어플리케이션을 사용하는 경우에는 경우에 따라서 DBCLIENT.DLL, STDVCL.DLL 파일을 윈도우 시스템 디렉토리에 복사하거나, BDE 나 SQL Link 를 설치할 필요가 있다.

- 인터넷 이용을 위한 DCOM 설정

DCOM 을 인터넷과 방화벽(firewall)을 넘어서 사용할 수 있도록 설정하기 위해서는 다음과 같은 사항을 고려해야 한다.

1. 서버와 클라이언트에서 모두 DcomCnfg.exe 유틸리티를 실행하여 인증 수준을 '없음'으로 설정한다.
2. 레지스트리의 내용을 다음과 같이 수정한다.

HKLM/Software/Microsoft/Rpc/Internet

PortsInternetAvailable="Y"

UseInternetPorts="Y"

Ports="3000-4000"

3. 135 번 포트를 열고 방화벽을 넘도록 한다.
4. IP 주소 번역을 Disable 한다.

DCOM 에 대한 HTTP 터널링(tunneling)은 NT SP4 에서부터 사용할 수 있게 되었다.
보다 자세한 내용은 마이크로소프트에서 제공하는 정보를 참고하기 바란다.

- 에러 메시지 정리

에러 메시지	원 인
DCOM not installed	DCOM 이 설치되지 않았다.
Server execution failed	1) 레지스트리의 EnableRemoteConnect 값이 N 으로 설정된 경우 2) 서버 어플리케이션이 실행되지 않은 경우 3) 서버가 로컬 드라이브에 존재하지 않는다. 4) 서버 객체에 접근하는 사용자의 권한이 미달된 경우 5) 레지스트리의 서버 패스가 지나치게 긴 경우 6) DCOMCNFG.exe 에서의 어플리케이션 위치 설정이 틀린 경우
Class not registered	서버 어플리케이션이 아직 등록되지 않음
RPC server is unavailable	1) RPCSS 가 서버에서 실행되지 않음 2) 레지스트리의 EnableRemoteConnect 값이 N 으로 설정된 경우 3) 잘못된 RemoteServer, ComputerName 4) TCP/IP 설정이 잘못된 경우
Interface not supported	1) 사용자 접근 레벨로 설정되지 않은 경우 2) Permission 이 Everyone 에게 허용되지 않은 경우 3) 레지스트리의 LegacyAuthenticationLevel 값이 설정되지 않은 경우

	4) Guest 계정이 disable 된 경우 5) 클라이언트가 클라이언트 기계에 인터페이스를 등록하지 않고 vtable 바인딩을 시도하는 경우
Access is denied	1) DCOM 보안 설정이 적절치 못한 경우 2) 서버 어플리케이션이 로컬 드라이브에 없는 경우

정 리 (Summary)

이번 장에서는 액티브 X 기술을 이용하여 어플리케이션이나 객체를 개발하면서 만날 수 있는 여러가지 문제점과 그 해결책에 대해서 알아보았다.
여기에 나열한 여러가지 팁 들보다 실제로 구현 과정에 들어가면 예상외로 많은 난관에 부딪히게 된다. MS 에서는 나름대로 분산 환경을 지원하기 위한 표준화 방안으로 DCOM 을 제시한 것이지만, 실제로 이를 활용하여 분산 환경을 구축하는 데에는 아직도 많은 어려움이 있으며, 개선해야 할 부분이 많은 것으로 생각된다.
그렇지만, 앞서 나가는 개발자가 되기 위해서는 이러한 문제점 들을 파악하고 나름대로의 해결책을 찾으려고 노력하는 자세가 필요할 것이다.