닷넷에는 자바 애플릿이나 혹은 액티브X 컨트롤처럼 웹 페이지에서 운용되는 컨트롤 기능이 있습니다. 이 기능은 자바 애플릿이나 액티브X 컨트롤보다 더 많은 기능들을 닷넷 프레임워크 차원에서 지원합니다. 이 장에선 닷넷에서 제공하는 스마트 클라이언트 관련 기능에 대한 설명과 델파이2005에서 활용하고 실제 응용프로그램에서 사용하는 방법에 대해서 설명합니다. 이 장은 이론들을 익히고 모든 실습을 완료 하는데 소요되는 예상 시간은 60분 정도 입니다.

'웹 기반' 응용프로그램

웹 이란 환경에 익숙해진 것은 90년 중반 이후 입니다.

지난 수년간 소프트웨어를 개발, 사용, 생성, 배포하는 방법들에 대해서 상당히 많은 진전이 있었습니다. 그리 오래된 얘긴 아니지만 클라이언트/서버 기술은 가장 큰 유행과도 같았습니다. 이런 환경에서 인터넷 네트워킹은 소프트웨어 분야를 모두 웹으로 전환하는 엄청난 파워를 자랑합니다. 새로 시작하는 프로젝트뿐만 아니라 기존의 클라이언트/서버 시스템도 모두 웹으로 전환하는 붐을 만들게 되었습니다.

이처럼 대부분의 소프트웨어 개발이 '웹 기반'으로 전환하게 된 데에는 그만한 이유들이 있습니다. 그 이유를 필자는 '씬 클라이언트(thin client)'의 장점이라고 말하고 싶습니다. 보통 씬 클라이언트라고 하면, 하드디스크나 주변장치 없이 최소한의 하드웨어와 네트워킹 기능만을 갖추고 대부분의 기능이 서버에서 수행되는 단말기를 말합니다. 웹 기반 시스템에서 브라우저는 바로 이런 '씬 클라이언트'의 역할을 하고 있습니다. 브라우저만 있으면 어떤 작업이든 할 수 있고 대부분의 데스크톱에서 사용할 수 있으며 클라이언트 컴퓨터의 상태에 영향을 주지 않습니다. 이것이 바로 웹 기반 애플리케이션의 가장 큰 장점입니다.

가장 먼저 꼽을 수 있는 것은 '배포 문제의 해결'입니다. 고전적인 클라이언트/서버 모델의 시스템에서 가장 큰 문제는 바로 클라이언트 애플리케이션을 배포, 설치, 업그레이드하는데 소요되는 비용이었습니다. 웹 기반 시스템은 중앙 집중적인 시스템으로 애플리케이션을 배포하고 관리하고, 기능을 개선하는 일을 쉽게 처리할 수 있습니다. 사용자 입장에서 볼 때 애플리케이션을 설치하거나 관리할 필요가 없고, 브라우저만 있으면 장소에 구애받지 않고 애플리케이션을 사용할 수 있어 '이동성'을 보장받을 수 있습니다.

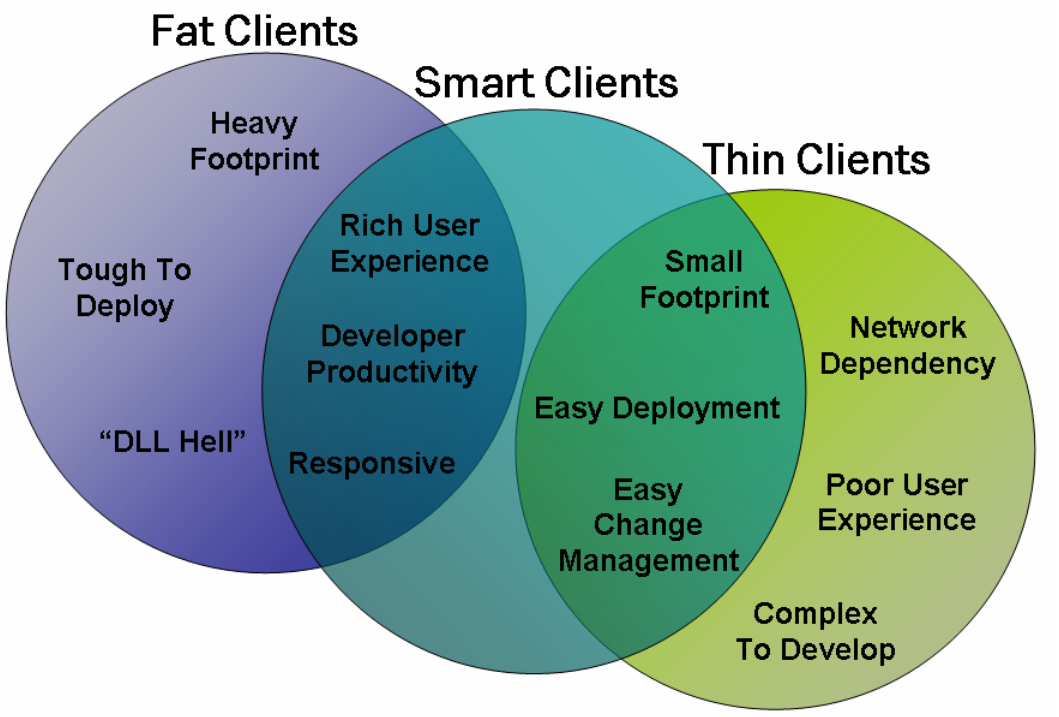
하지만 모든 기술이 그러하듯 '웹 기반' 시스템이 장점만을 가진 것은 아닙니다. 업무가 복잡해지고 사용자의 요구사항이 증대되면서 온라인과 오프라인에 관계없이 풍부한 기능 등이 필요하게 되었습니다. 사용자와의 더욱 다양한 상호작용과 미려한 사용자 인터페이스를 제공해야 할 필요성이 생겼고, 브라우저가 제공하지 못하는 추가적인 기능을 제공할 필요성이 높아진 겁니다.

스마트 클라이언트

스마트 클라이언트 애플리케이션은 일반적인 특징을 가지고 개발된 애플리케이션을 지칭 합니다. 그러한 특징을 살펴보면 다음과 같은 것들이 있습니다.

- 애플리케이션이 실행되는 장치와 밀접하게 동작하여 장치의 이점을 살리고 있습니다.
- 사용자에게 더 많은 기능과 정보를 제공하기 위해서 XML 웹 서비스를 이용하고 있습니다.
- 온/오프라인으로 직접 연결되지 않을 때도 전혀 불편함을 주지 않습니다. 더불어서 개인적 휴대용 장비인 PDAs나 모바일 폰으로도 사용되고 있습니다.

스마트 클라이언트 애플리케이션은 Java, Macromedia등 여러 기술들을 사용해서 구현될 수 있습니다. 그러나 볼랜드는 Delphi2005를 이용해서 더 빨리 그리고 쉽게 애플리케이션을 만들 수 있게 하고 있습니다. Delphi2005는 개발자들이 고객의 요구에 맞게 개발을 쉽게 할 수 있도록 닷넷의 모든 기능과 MS 닷넷 프레임워크에 추가적인 볼랜드만의 프래임워크를 포함하여 개발비용, 기간을 단축시키고, 기존의 환경에 더욱 최적화 할 수 있도록 개발 환경을 제공 하고 있습니다.



[그림 스마트 클라이언트]

위 그림은 스마트 클라이언트가 어떻게 양쪽의 장점만을 취하는지 나타내고 있습니다.
팻 클라이언트(Fat Client) 는 씬 클라이언트에 대비되는 말로, (좁은 의미에서) 클라이언트의 자원을 적극 활용하는 코드를 배포해 HTML과 브라우저가 제공하지 못하는 기능을 제공하는 것으로 정의할 수 있습니다. 팻 클라이언트는 '웹 기반 시스템'의 중앙 집중적인 시스템의 단점과 HTML이 가진 원천적인 한계을 극복할 수 있는 대안 중의 하나가 될 수 있습니다.
예를 들어, 웹 기반의 멀티미디어 서비스를 위해 널리 사용되는 플래시가 좋은 예가 될 수 있습니다. 플래시 자체가 브라우저에 추가되는 컨트롤이지만, 플래시는 단순한 컨트롤을 넘어 다양한 멀티미디어 기능을 제공하는 '하나의 플랫폼'입니다. 다른 예로, 시중의 은행이나 증권에서 제공하는 인터넷 뱅킹 전용 프로그램과 증권 정보 시스템등을 들 수 있습니다. 클라이언트에 설치되는 프로그램이면서 웹으로 제공하지 못하는 다양한 기능과 빠른 UI를 제공합니다. 팻 클라이언트는 대부분의 작업을 클라이언트에서 수행하도록 할 수 있습니다. 특히, 웹 기반 애플리케이션처럼 UI에 대한 데이터나 코드를 주고받을 필요가 없기 때문에 서버에 부담되는 가중이 적을 뿐만 아니라 네트워크의 트래픽을 상당히 많이 줄일 수 있습니다. 매번 서버로부터 UI에 필요한 리소스를 받아올 필요도 없고 정확히 필요한 데이터만을 주고받으면 되기 때문이다(웹 페이지를 구성하는 데이터 중에서 UI 구성요소를 제외한, 실제 데이터의 양은 수십 분의 일 밖에 안 된다는 점을 고려해야 합니다.).
팻 클라이언트는 로컬 클라이언트에 저장됩니다. 이것은 풍부한 사용자 인터페이스와 경험을 제공할 수 있다는 것입니다. 또한 직접적인 하드디스크 및 직접적인 애플리케이션 사용으로 인해서 즉각적인 응답을 얻으므로 개발자에게 생산성을 증대시켜 줍니다.
다른 한편으로 단점도 가지는데 배포와 업그레이드가 그것입니다. 버전관리가 제대로 이루어 지지 않으면 "DLL Hell"을 경험 하게 됩니다. 서로 참조되는 DLL끼리 버전이 맞지 않아서 발생하는 다양한 에러를 발생하게 되는 것입니다.

반면 "thin" 또는 브라우저 베이스인 애플리케이션은 많은 장점들이 있습니다. 웹 베이스 애플리케이션은 인터넷 브라우저를 통해서 실행 됩니다. 씬 클라이언트 애플리케이션은 쉽게 많은 데스크탑에 배포되고 유지될 수 있습니다. 실행파일의 크기도 작고 씬 클라이언트는 클라이언트 컴퓨터에 거의 영향을 미치지 않습니다. 이것은 IT 기술에 대단한 잇점이라 하겠습니다.
그러나 씬 클라언트도 완벽하다고 볼 수 없습니다. 씬 클라이언트는 브라우저에 의해 실행되기 때문에 브라우저의 한계를 안고 있습니다. 아무리 좋은 인터페이스를 구현하고 싶다고 하더라도 브라우저에서 지원하지 않는 기능이라면 구현을 할 수 없는 것입니다. 또 온라인이 항상 유지될 수 없는 환경에서도 똑 같은 기능을 수행하길 바란다면 씬 클라이언트는 적합하지 않습니다. 씬 클라이언트는 온라인상에서만 기능을 유지 할 수 있다는 문제점을 가지고 있습니다. 이런 이유로 사용자는 팻 클라이언트 보다 더 불편하고 덜 떨어지는 기능을

가진 씬 클라이언트의 UI를 경험하게 됩니다. 또한 화면을 표시하기 위해서 매번 네트워크 트래픽을 유발합니다.
위 그림에서 스마트 클라이언트는 잘 표현되어 있습니다. 양쪽에 장점을 잘 취해서 결합된 형태입니다.
위 그림을 좀 더 자세하게 표로 만들어 보면 아래와 같습니다.

문제점	Fat 클라이언트	Thin 클라인언트	스마트 클라아인트
풍부한 기능과 응답의 필요	OK	No (기술의 제약점)	비교적 OK
손쉬운배포/관리/업데이트	OK (추가적인시간과/자원요구)	OK	OK
오픈라인일 때, 사용자의 작업	OK (복잡한 Sync코드 필요)	No	OK
기업내의 기존 데이터와 연동/재사용 가능	백엔드 시스템은 재사용불가	재사용이 어려운 연결에 대한 투자	OK(WebSerivce)

MS는 자사의 닷넷 코드 배포 기술과 XML 웹 서비스 기술을 이용하는 클라이언트를 가리켜 스마트 클라이언트(smart client)라고 합니다.
주의 할 것은 닷넷에 스마트 클라이언트를 구현을 위한 기능(feature)은 따로 있는 것이 아니라는 점입니다. 좀더 구체적으로 얘기하면 액티브X 컨트롤 같은 경우는 브라우저가 컨테이너 역할을 하는데 여기서 말하는 닷넷 스마트 클라이언트의 실질적인 컨테이너는 CLR이고 브라우저는 단지 어셈블리 참조를 하는 애플리케이션이 됩니다. 즉, HTTP로 어셈블리가 배포될 뿐 기존의 액티브X 컨트롤처럼 브라우저만을 위해 따로 존재하는 코드 배포 기술이 아니라 닷넷 CLR이 가진 원래 기능입니다. 어떻게 보면 스마크 클라이언트라고 하면 배포를 하기 위한 기술이지 클라이언트 자체를 의미 하는 것은 아닙니다.

XML

XML 웹 서비스들은 모든 스마트 클라이언트 애플리케이션을 위해서 사용됩니다. XML 은 이미 알고 있듯이 언어 및 플랫폼에 관계없이 인터넷 애플리케이션을 프로그래밍 하기 위한 표준 프로토콜 입니다. XML 웹 서비스는 단순히 데이터를 공유하는 것뿐만 아니라 다른 애플리케이션이 어떻게 구성 되어 있든 상관없이 애플리케이션를 호출 할 수 있습니다.
XML 웹 서비스는 표준 웹 프로토콜인 HTTP, XML로 구성됩니다. 이 서비스를 위한 기술적으로 도입된 것은 다음과 같은 것들이 있습니다.

DISCO (Discovery) 와 UDDI (Universal Description, discovery, Integration).
SOAP (Simple Object Access Protocol) 와 explicit serialization (HTTP + XML description),
WSDL (XML Web services Description Language)

XML 웹 서비스는 분산 애플리케이션에 새로운 모델을 제시하고 있습니다. 왜냐하면 오픈되어 있고, 인터넷 표준을 가지므로 어떠한 언어, 어떠한 플랫폼에서든 통신하는 것이 가능하기 때문입니다. 스마트 클라이언트 애플리케이션은 웹 서비스를 호출할 수 있고, 호출된 대상은 가상의 데이터 소스와 직접적으로 연결될 수 있습니다. 이러한 것은 불필요한 데이터의 감소와 변경을 통해서 문서 및 형식화된 데이터를 받을 수 있습니다.
이는 또한 UDDI, WSDL를 통해서 인터넷/인트라넷에 애플리케이션 로직을 노출시킬 수 있습니다. SOAP은 다양한 운용시스템에 있는 애플리케이션으로부터 웹 애플리케이션을 쉽게 사용할 수 있도록 합니다. 웹 서비스는 서비스 내부구현에 대해서 전혀 알 필요가 없습니다. 즉, 클라이언트는 내부가 어떻게 구현이 되어 있는지 알 필요 없이 공개된 메소드를 사용하기만 하면 됩니다.

Delphi2005

Delphi 2005는 실질적으로 스마트 클라이언트 개발자를 포함해서 모든 애플리케이션 개발자들을 위한 강력한 툴입니다. Delphi, C#등 원하는 언어로 프로그램을 할 수 있고, 닷넷 프레임워크를 통해 닷넷 팻 클라이언트 애플리케이션 개발과 Win32용 애플리케이션 개발을 위한 최적의 디자이너와 개발자 환경을 가지고 있습니다.
(* 앞으로 Delphi2006버전에 포함될 컴팩 프렘워크를 통한 스마트 디바이스 애플리케이션 개발도 가능할 것입니다.)
또한 표준 HTML 애플리케이션과 모바일 웹 애플리케이션을 위한 폼 또한 최적화 할 수 있습니다. 성능카운터, 메시지 큐, SQL 서버 데이터베이스등 다양한 추가 기능등도 제공합니다.

윈폼

Delphi2005는 닷넷의 윈 폼 애플리케이션을 쉽게 만들 수 있습니다. 윈폼에서 제공하는 모든 컨트롤 들은 쉽게 몇줄의 코딩으로 확장 될 수 있습니다. 여기에는 도킹/언도킹을 포함해서, 메인메뉴, 컨텍스트 메뉴등도 있습니다. 그리고 더 많은 이미지 포맷을 지원하는 컨트롤과 함께 GDI+ 의 장점을 최대한 활용 할 수 있습니다.

배포

닷넷 프레임워크는 애플리케이션을 배포하기 위한 옵션으로 다음과 같은 것을 제공하고 있습니다.

첫째, XCOPY 또는 FTP 가 그것 입니다. CLR 애플리케이션은 닷넷 애플리케이션이 실행되기 위해 필요한 별도의 파일들을 실행되는 폴더에서 찾아 활용 합니다. 그렇게 때문에 어떠한 레지스트로도 등록할 필요가 없고 단지 복사하거나 FTP 를 통해서 파일을 업로드 하는 수준만으로 배포 작업이 끝납니다. 물론 인스톨 프로그램을 사용하여 배포하는 건 두말할 필요 없이 가능한 얘깁니다.

보안

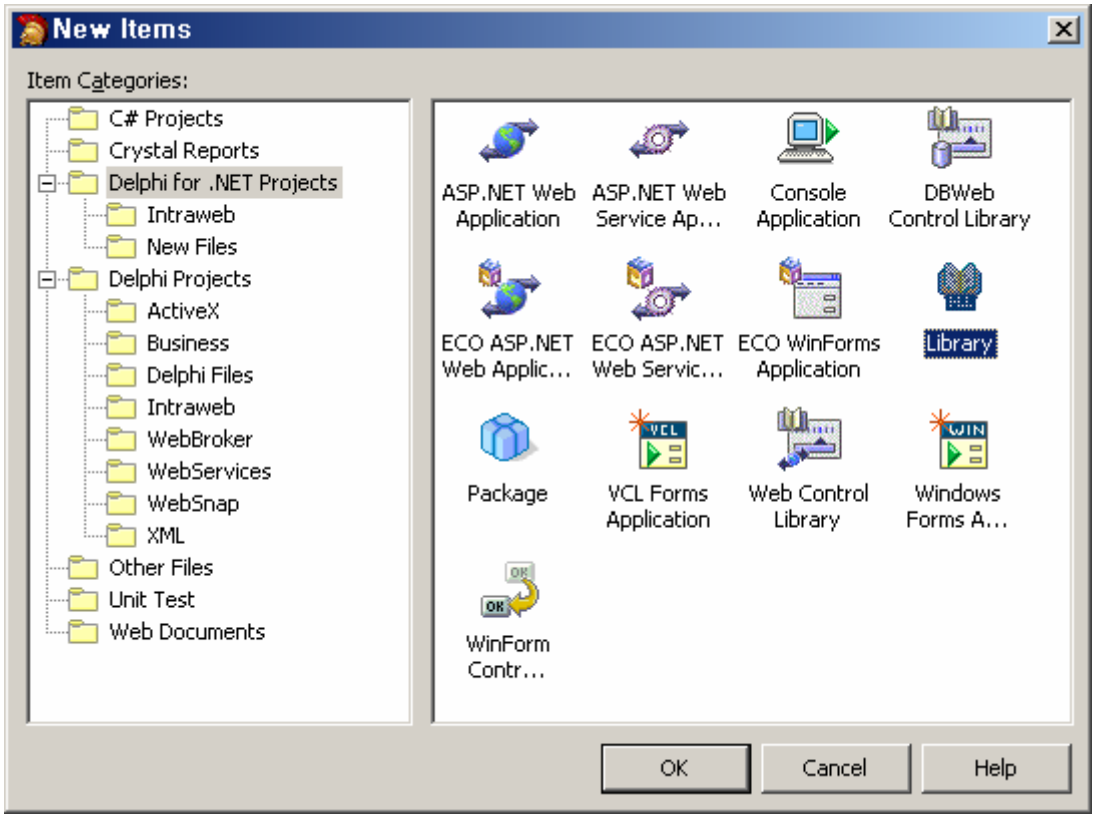
닷넷의 관리 코드가 시작 될 때 언제라도 프레임워크는 코드 특성 및 실행환경을 조사하게 됩니다. 만일 실행환경과 맞지 않을 때 실행이 되면 프레임워크가 실행을 강제로 중단 시킵니다. 이러한 실행 모델은 코드 수행환경설정과 실행 환경(로컬, 네트워크 공유, URL등)이 무엇인지에 따라서 결정됩니다. 전형적으로 로컬컴퓨터에서 실행된다면 “완전 신뢰” 권한을 가지게 되어 별다른 제한 없이 응용 애플리케이션이 실행 됩니다. 그러나 네트워크 공유상에서 실행을 하게 되면 별도의 인증 코드를 요구하게 되고 권한도 내려 갑니다. 이 환경에서 인트라넷에 권한이 제한되어 있으면 코드가 실행될 때 인트라넷 권한이 지정된 이상의 기능을 수행 할 수 업게 됩니다.

실습

기본적인 정보를 바탕으로 간단한 스마트 클라이언트를 만들어 보고 이를 테스트 해보겠습니다. 아래 순서대로 프로젝트를 만들고 폼을 추가하여 기본적인 닷넷 **DLL**을 만듭니다.

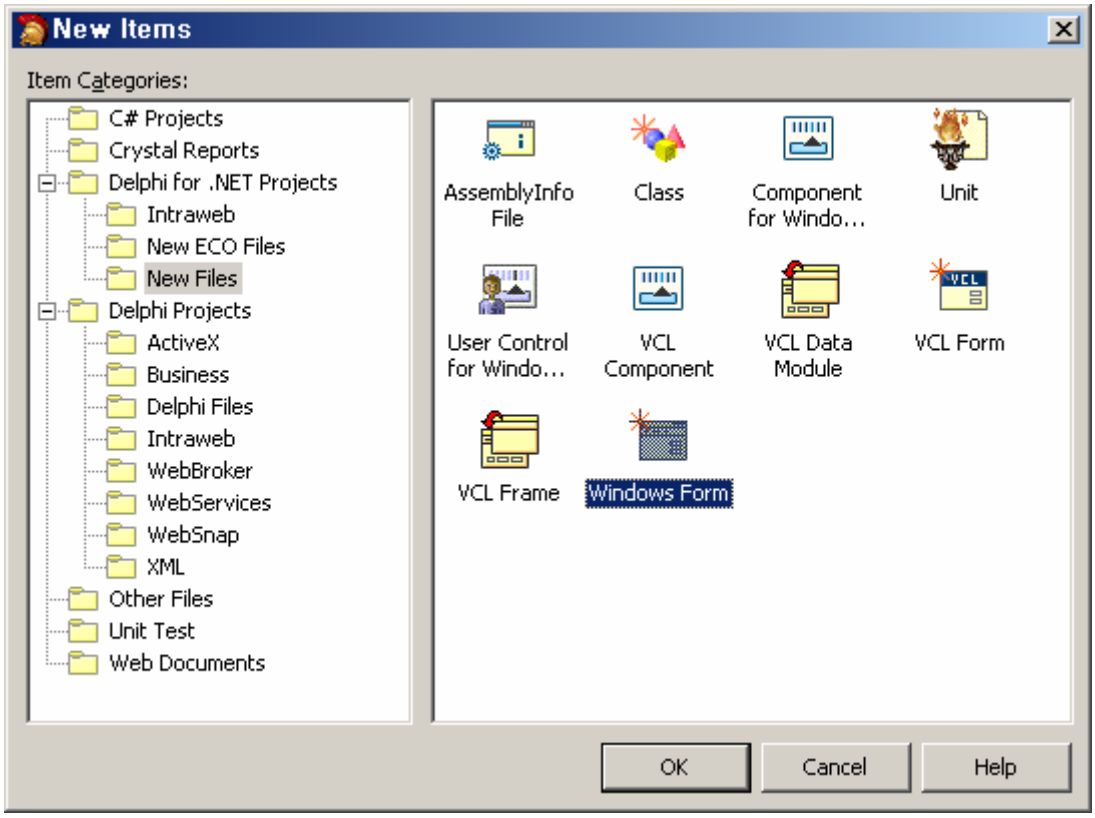
(가) 서브 모듈 만들기

[1]



Delphi2005에서 메뉴 File>New>Other를 선택하고, Library 프로젝트를 생성 합니다.

[2]

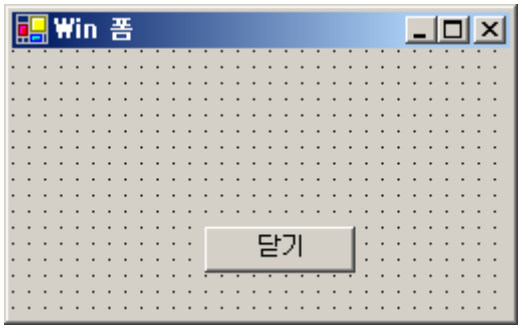


Delphi for .NET Project - New Files 에서 Windows Form을 추가합니다.

[3]

Win Form Text 속성에 'Win 폼'이라고 입력합니다.

[4]



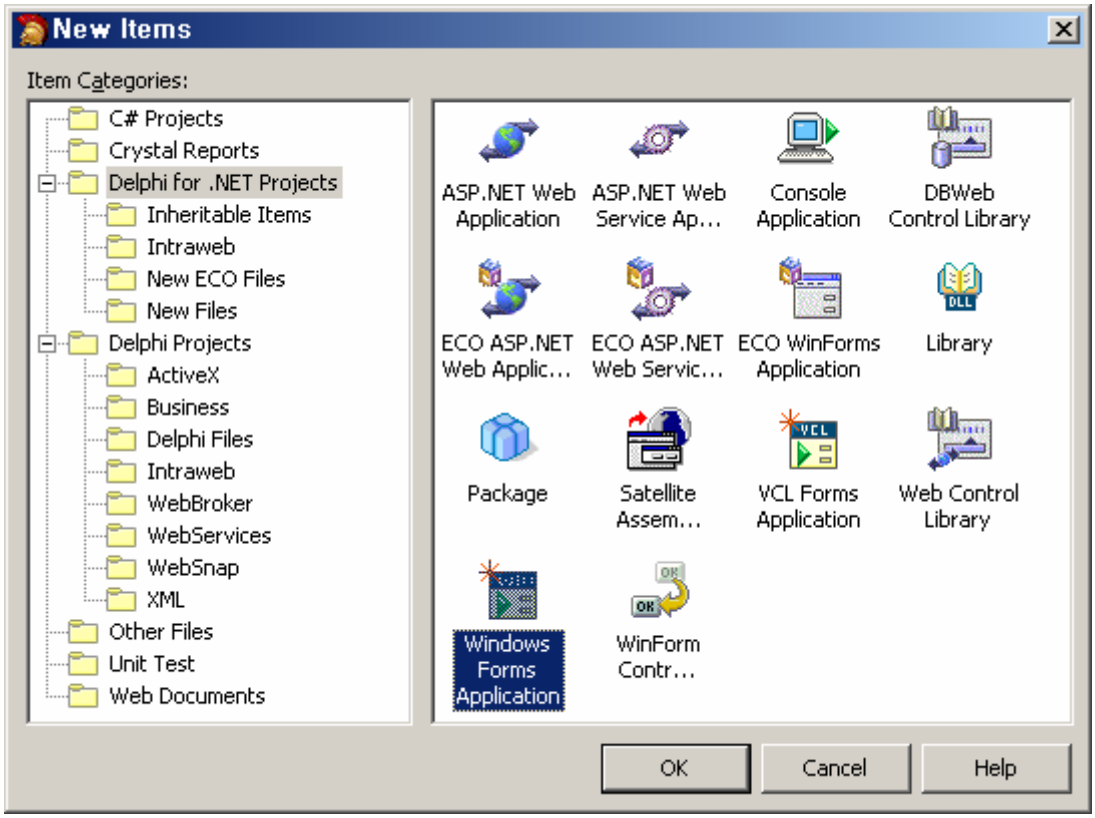
Win Form의 크기를 적절하게 줄이고 Button 컨트롤을 올려놓고 Button Text 속성에 '닫기'라고 입력합니다.

[5] 소스를 저장하고 컴파일해서 dll 파일을 만듭니다.

[6] 이렇게 만들어진 dll 파일은 IIS 웹 서비스가 가능한 경로에 복사합니다.

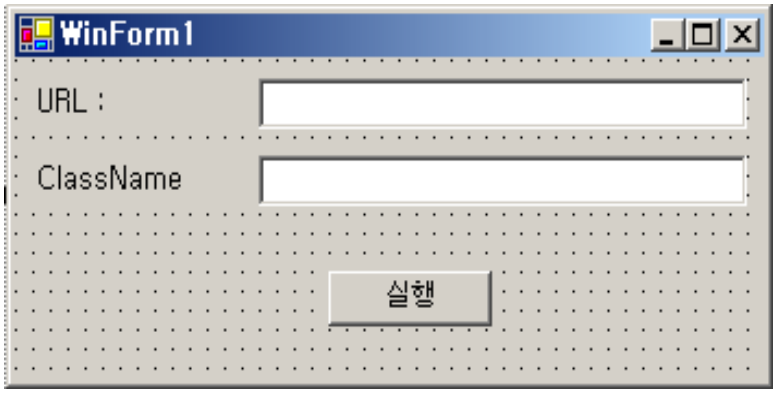
(나) 메인 프로그램 만들기

[1]



Delphi2005에서 메뉴 File>New>Other를 선택하고, Windwos Forms Application선택하여 프로젝트를 생성 합니다.

[2]



TextBox 2개, Label 2개 1개의 Button을 폼에 올려 놓고 각각의 Text 속성을 다음과 같이 입력합니다

Label1.Text := 'URL :';
Label2.Text := 'ClassName';

Button1.Text := '실행';

TextBox1 Name : txtUrl
TextBox2 Name : txtClassName

txtUrl.Text := '';
txtClassName.Text := '';

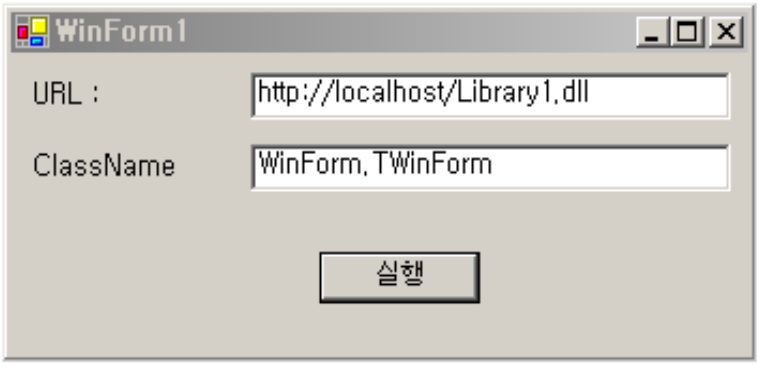
[3]
Button1의 Click 이벤트 핸들러를 아래 소스를 보고 작성합니다.

다음은 전체 소스입니다.

```
1   unit WinForm1;
2
3   interface
4
5   uses
6     System.Drawing, System.Collections, System.ComponentModel,
7     System.Windows.Forms, System.Data, System.Reflection;
8
9   type
10    TWinForm1 = class(System.Windows.Forms.Form)
11    Designer Managed Code
29    strict protected
30      /// <summary>
31      /// Clean up any resources being used.
32      /// </summary>
33      procedure Dispose(Disposing: Boolean); override;
34    private
35      { Private Declarations }
36    public
37      constructor Create;
38    end;
39
40    [assembly: RuntimeRequiredAttribute(TypeOf(TWinForm1))]
41
42  implementation
43
44  {$AUTOBOX ON}
45
46  Windows Form Designer generated code
114
115 procedure TWinForm1.Dispose(Disposing: Boolean); ...
124
125 constructor TWinForm1.Create; ...
136
137 procedure TWinForm1.Button1_Click(sender: System.Object; e: System.EventArgs);
138 var
139   app: Assembly;
140   frmType: System.Type;
141   frmObj: System.&Object;
142   frmSub: Form;
143 begin
144   app := Assembly.LoadFrom(txtUrl.Text);
145   frmType := app.GetType(txtClassName.Text);
146
147   if frmType <> nil then
148   begin
149     frmObj := Activator.CreateInstance(frmType);
150     frmSub := frmObj as Form;
151     frmSub.Show;
152   end
153   else
154   begin
155     MessageBox.Show('선택된 폼이 존재하지 않음');
156   end;
157 end;
```

System.Reflection 추가

(다) WinForm 불러오기



참고로 XP를 OS로 사용하고 있다면 보안(방화벽) 설정 중에 80포트를 사용할 수 있도록 설정을 해야 합니다. 필자도 당황한적이 있습니다.^^

내용의 상당부분은 http://msdn.microsoft.com/smartclient/ 에서 참고 하였습니다. 관심이 있는 분들은 사이트를 꼭 참고하시기 바랍니다. 물론 여기엔 델파이에 대한 내용은 없지만 닷넷의 이점은 언어와 관계가 없어 쉽게 적용이 가능 합니다.