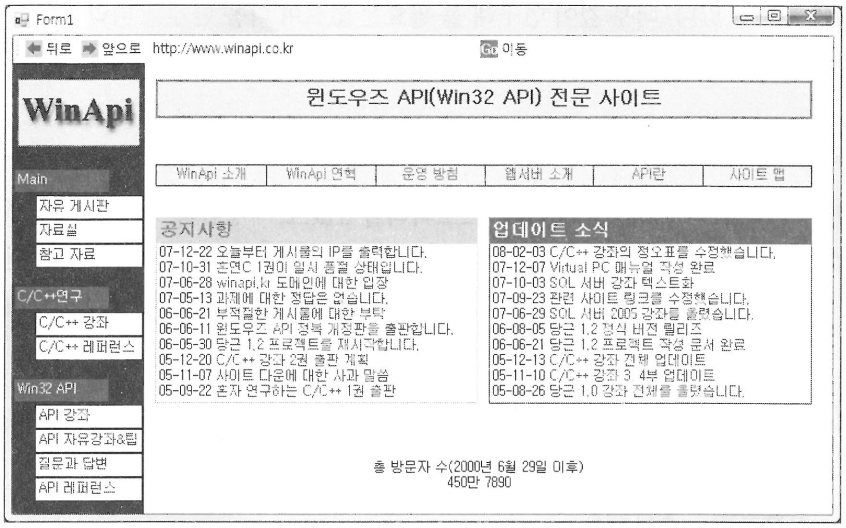
툴바를 위에 배치하고 앞뒤 이동 및 주소 입력을 위한 텍스트 박스, 이동 버튼을 배치했다. 비록 잘 그리지는 못하지만 툴바에 이미지도 그려 넣었다. WebBrowse 컨트롤의 Dock 프로퍼티를 Fill로 설정하여 폼의 남는 영역을 모두 차지하도록 했다. 툴바의 텍스트 박스에 주소가 입력되면 Navigate 메서드로 주소를 넘기기만 한다. 실행 직후 텍스트 박스의 프로퍼티에 설정되어 있는 기본 웹 사이트로 이동한다.




웹 문서를 읽어와 출력하는 과정은 상상을 초월할 정도로 복잡하다. 네트워크에 접속해야 하고 HTML 문서를 읽어와 분석하고 이미지와 부가 파일들까지 다운로드 받아 적절한 위치에 출력해야 한다. 또한 링크를 누르면 해당 링크로 이동하고 클라이언트 스크립트를 해석하여 실행하고 플러그인이나 ActiveX를 호출하기도 한다. 이런 복잡한 동작을 하는 웹 브라우저를 컨트롤 하나만 배치하면 쉽게 만들 수 있다. 정말 간단하지 않은가? 이게 바로 컴포넌트의 매력이며 객체 지향의 강력함이다.

## 2.6 MDI

MDI(Multiple Document Interface)는 폼 안에 또 다른 자식 폼을 여러 개 열 수 있는 형태의 응용 프로그램이다. MDI를 사용하는 대표적인 응용 프로그램으로 엑셀이나 포토샵 등이 있는데 여러 개의 워크시트나 사진을 동시에 열어 놓고 작업할 수 있다. 반대 개념은 SDI인데 한 번에 하나의 문서만 열 수 있으며 메모장이 대표적이다. MDI는 동시에 여러 개의 문서를 열어 놓고 작업할 수 있어 기능적으로는 SDI보다 더 우위에 있지만 초보자에게 혼란을 준다는 이유로 요즘은 거의 사용되지 않으며 마이크로소프트는 MDI를 사용하지 말 것을 공식적으로 권고하고 있다.

윈도우즈를 처음 사용하는 사람들은 윈도우 안에 또 다른 윈도우가 있다는 사실을 얼른 이해하지 못하며 심지어 익숙한 사람들조차도 MDI는 그다지 편리하지 않다고 한다. 그래서 요즘은 문서 당 하나

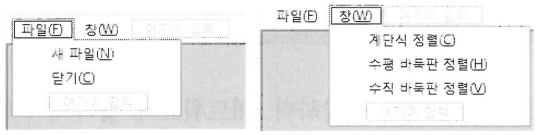
의 윈도우를 여는 DOI가 더 유행하고 있으며 워드를 필두로 하여 아래 한글 등도 모두 DOI(Document Oriented Interface)로 전환했다. 비주얼 C++도 6.0까지는 MDI를 사용했으나 7.0 이후 버전부터는 독특한 다른 구조를 사용하고 있다.

그러나 아직까지도 MDI가 유용한 경우가 남아 있어 대부분의 개발 툴은 MDI를 지원하며 닷넷도 MDI를 잘 지원한다. 그러나 전반적으로 MDI를 회피하는 추세여서 가장 기본적인 기능만 지원할 뿐이며 생색만 내는 정도에 그치고 있다. 너무 깊이 연구해 볼 필요는 없으며 다음 단계를 따라 간단한 예제 하나만 만들어 보자.

### 예제 MDIForm

1. 새 프로젝트를 만든 후 Form1의 IsMdiContainer 프로퍼티를 true로 지정한다. 이 프로퍼티를 true로 지정하면 MDI 프레임 윈도우로 사용되며 배경색은 훨씬 더 짙은 회색이 된다. 프레임 윈도우는 내부에 자식 폼 여러 개를 관리해야 하므로 자식 폼보다는 커야 한다. 충분히 크기를 늘리도록 하자.

2. 자식 창을 관리하기 위한 메뉴를 만든다. MenuStrip 컴포넌트를 배치하고 파일과 창 메뉴를 만든다.




새로운 문서를 만들고 닫는 명령과 자식 창을 정렬하는 명령들이다. 물론 실제 프로젝트에서는 응용 프로그램 고유의 메뉴 명령들도 같이 만들어야 할 것이다.

3. 프로젝트/구성 요소 추가 메뉴 항목을 선택한 후 새로운 폼을 하나 더 추가한다. Form2라는 이름으로 빈 폼이 생성될 것이다. 여기에 텍스트 박스를 배치하고 Multiline 프로퍼티를 true로, Dock 프로퍼티를 Fill로 지정한다. 간단한 텍스트 편집 창 하나를 만든 것인데 실제 프로젝트에서는 작업 내용에 맞게 컨트롤을 배치하고 코드를 작성하면 된다.

4. 메인 폼에 자식 폼을 관리하는 명령들을 작성한다. 메뉴 항목들을 차례대로 더블클릭하여 Click 이벤트 핸들러를 생성하고 다음 코드를 작성한다.

```
using System; ~ using System.Windows.Forms;

namespace MDIForm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void 새파일NToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 Child = new Form2();
            Child.MdiParent = this;
            Child.Show();
        }

        private void 닫기CToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form Child = ActiveMdiChild;
            if (Child != null)
            {
                Child.Close();
            }
        }

        private void 계단식정렬CToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LayoutMdi(MdiLayout.Cascade);
        }

        private void 수평바둑판정렬HToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LayoutMdi(MdiLayout.TileHorizontal);
        }

        private void 수직바둑판정렬VToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LayoutMdi(MdiLayout.TileVertical);
        }
    }
}
```

새로운 자식 폼을 만들 때는 new 연산자로 폼 객체를 하나 만들고 MdiParent에 프레임 윈도우를 대입하면 된다. Show 메서드로 자식 폼을 보이면 프레임 윈도우 안에 나타날 것이다. 닫을 때는 프레임의 ActiveMdiChild 프로퍼티로 현재 활성화된 폼을 구한 후 Close 메서드를 호출한다. 자식 창을 정렬할 때는 LayoutMdi 메서드를 호출하되 인수로 계단식, 수평 바둑판, 수직 바둑판 등의 옵션을 전달한다.


MDI 프로그램의 창 메뉴에는 현재 열려진 문서 창의 목록이 나타나며 이 메뉴를 통해 자식 창 중 하나로 전환할 수 있다. 닷넷은 이 기능을 자동으로 구현하는 기능도 제공하는데 안타깝게도 구형 MainMenu 컴포넌트에만 이 기능이 있고 닷넷 2.0에 새로 추가된 MenuStrip 컴포넌트에서는 지원하지 않는다.