나만의 윈도우 라이브 가젯 만들기
StockViewer 가젯 만들기

목차

소개

지난 연재를 통해서 가젯에서 사용하는 자바스크립트와 가젯 API에 대해서 자세하게 배웠다. 이번 시간은 지금까지 배운 지식을 토대로 StockViewer 가젯을 제작하는 과정에 대해서 다룬다. 홈페이지의 html 데이터를 가공해서 사용하는 방법과 StockViewer 가젯 제작에 있어 필자가 어려움을 겪었던 부분에 대해서 설명한다.

연재 가이드

운영체제: 윈도우 2000/XP
개발도구: Editplus, IE7 or FireFox
기초지식: Javascript, HTML, CSS
응용분야: Windows Live Gadget 프로그램, AJAX 프로그램

연재 순서

2007. 02 이상한 나라의 자바스크립트

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

2007. 03 체스 기보 뷰어 만들기
2007. 04 Hello, World 가젯 만들기.
2007. 05 StockViewer 가젯 만들기

필자소개

신영진 pop@jiniya.net, http://www.jiniya.net
시스템 프로그래밍에 관심이 많으며 다수의 보안 프로그램 개발에 참여했다. 현재 데브피아 Visual C++ 섹션 시삽과 Microsoft Visual C++ MVP로 활동하고 있다. 최근에는 SpiderMonkey를 임베딩 시켜서 사용하는 것에 관심이 많다.

필자 메모

요즘은 여러 벤더들에서 자사 플랫폼이나 기술을 홍보하는 목적으로 컨테스트를 많이 개최한다. 이러한 컨테스트는 벤더 입장에서는 홍보 효과와 동시에 자사 플랫폼 기술자를 늘리기 때문에 이익이고, 개발자 입장에서는 새로운 플랫폼에 대한 지식과 함께 선정되는 경우에는 물질적인 보상까지 받을 수 있기에 이익이다. 서로 윈-윈 하는 게임인 셈이다.

안타까운 현실은 이러한 컨테스트에 참여하는 개발자가 많지 않다는 점이다. 컨테스트에 참여하지 못하는 가장 큰 이유는 보통 시간과 실력이다. 현업 개발자 입장에서는 시간이 부족하고 예비 개발자인 학생들 입장에서는 실력이 부족하다고 느끼는 것이다. 하지만 사실 컨테스트에 참가하기 위해서는 많은 시간도 대단한 실력도 필요 없다. 가장 중요한 것이 있다면 무엇인가를 하겠다는 열정이다.

Introduction

개인화 페이지에 추가할 수 있는 컨텐츠는 다양하다. 뉴스에서 시작해 연예인들의 가십 거리, RSS 피드, 스포츠 경기 결과, 일정 관리에 이르기 까지 너무나도 많은 것들이 있다. 그렇다면 여러분들은 자신의 개인화 페이지를 만든다고 했을 때 가장 먼저 무엇을 배치하고 싶은가? 작년부터 간간히 주식 투자를 하고 있는 필자는 주식 정보를 보여주는 가젯을 가장 먼저 설치하고 싶었다. HTS와 포탈 사이트 모두 매번 찾아 들어가기가 번거로웠기 때문이다.

이번 시간에 우리가 살펴볼 StockViewer는 코스피/코스닥에 상장된 기업의 주가를 실시간으로 보여주는 가젯이다(<그림 1> 참고). 사용자는 관심 있는 기업을 검색해서 등록, 삭제할 수 있다. 관심 기업으로 등록된 회사의 주가가 가젯의 메인 화면에 실시간으로 나타난다. 가젯을 설치해서 사용해 보고 싶은 독자들은 http://www.jiniya.net/gadget/Stock/StockViewerGadget.xml을 live.com에 설치하면 된다.

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

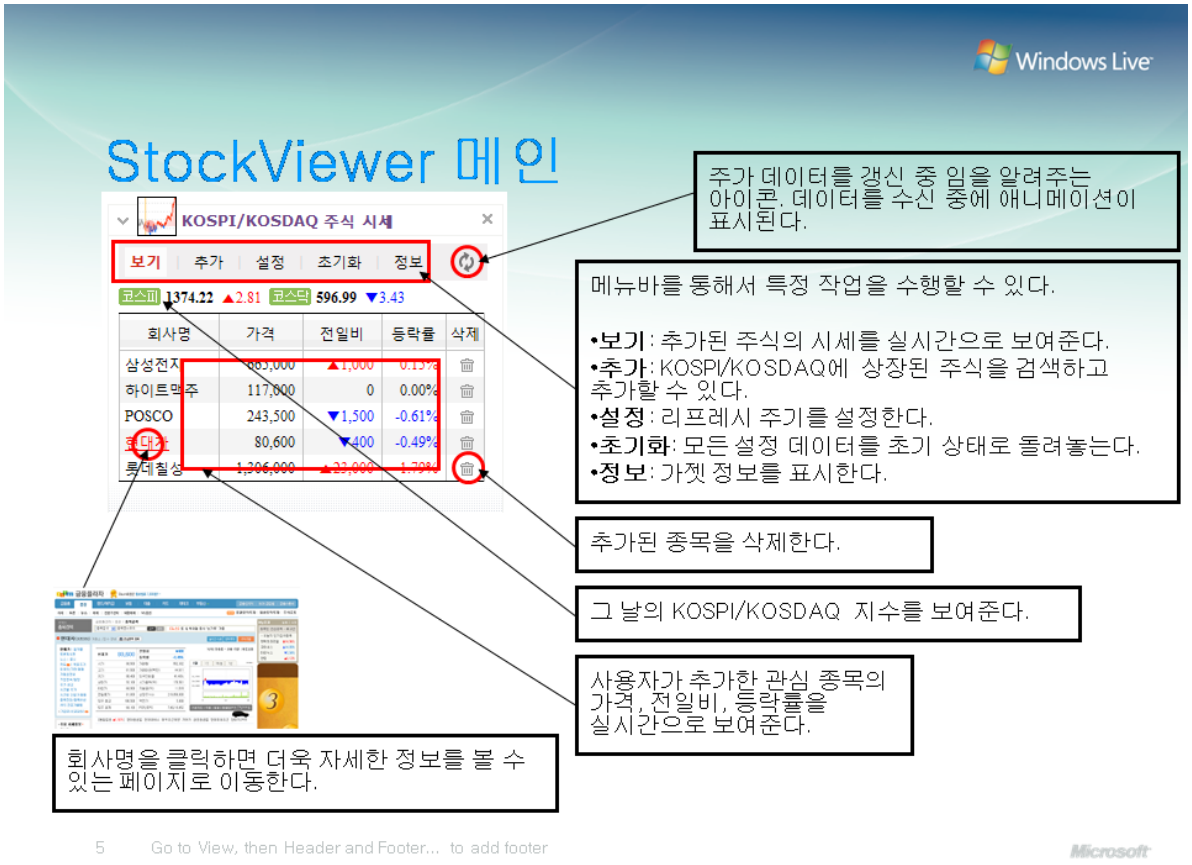


그림 1 StockViewer 메인 인터페이스 설명

주가를 어디서 읽어올까?

주식 가젯을 만들기 위해서 가장 먼저 생각해야 할 것은 주식 데이터를 가져올 곳이다. 주가 정보는 방대하기도 하고 실시간으로 변하는 데이터이기 때문에 개인이 직접 취급할 수 없다. 아직까지 국내에서는 가젯에서 바로 쓸 수 있는 XML 형태로 잘 가공된 주가 데이터를 제공해 주는 곳은 없다. 그나마 잘 가공된 데이터를 제공하는 곳이 포탈의 증권 페이지다.

포탈의 증권 페이지는 html 형태로 되어 있기 때문에 우리가 필요한 정보를 찾기 위해서는 별도로 페이지를 파싱하는 작업이 필요하다. 우리가 원하는 정보가 있는 페이지들은 대부분 CGI가 기계적으로 생성한 것들이기 때문에 형태가 변하는 일이 적고 분리해 내기가 쉽다. 하지만 페이지를 가공하는 업체 쪽에서 이러한 작업을 막기 위해서 주기적으로 페이지를 변경하는 일을 하는 경우에는 그때마다 파싱과 관련된 코드를 수정해 주어야 한다.

예전에 필자는 네이버의 사전을 기반으로 단어장 프로그램을 제작한 적이 있었다. 그 때 이러한 페이지 갱신 빈도 때문에 상당히 곤혹을 치렀다. 이런 이유로 이번에는 다음 페이지를 파싱해서 사용하기로 했다. 다음 페이지의 html이 좀 더 간단하다는 장점도 있었다.

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

회사 코드 정리

페이지를 파싱하기 전에 먼저 해야 할 작업은 각각의 회사의 주가 정보를 가지고 있는 페이지의 URL을 알아내는 작업이다. 아래는 다음 증권에서 제공하는 회사별 주가 정보를 제공하는 페이지 URL이다.

삼성전자 http://stock.finance.daum.net/search/quote/s2010000.html?code=005930
삼성전자우 http://stock.finance.daum.net/search/quote/s2010000.html?code=005935
LG전자 http://stock.finance.daum.net/search/quote/s2010000.html?code=066570

금방 알 수 있듯이 회사 정보를 얻기 위해서는 회사에 대한 고유 코드를 알아야 한다. s20100000.html의 인자로 넘어가는 code 변수에 대한 값이 그것이다. 주식 시장에 상장된 회사는 엄청나게 많다. 이것들을 하나씩 검색해서 알아내는 방법은 비효율 적이다. 다음 증권의 전종목시세 페이지(<화면 1> 참고)를 이용하면 회사명에 대응하는 코드를 쉽게 구할 수 있다.

화면 1 전종목시세

첫 눈에 봐도 회사가 얼마나 많은지 알 수 있다. 처음에 필자는 이 것들을 손으로 일일이 분석했다. 회사명을 복사해서 메모장으로 옮기고 해당 회사 페이지로 들어간 다음 코드 값을 복사해서 메모장으로 붙여 넣는 작업이었다. 십 분 정도를 이렇게 작업했다. 진행 과정은 무척 더뎠고, 저 많은 회사 코드를 다 알아내는 과정이 가젯을 만드는 시간보다 더 오래 걸릴 것 같았다. 뭔가 해결책이 필요했다. 필자가 생각한 방법은 간단한 파이썬 스크립트(<리스트 1> 참고)를 사용해서 페이지에서 데이터를 자동으로 추출하는 것이었다.

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

리스트 1 회사명과 코드를 추출하는 스크립트

```
import re

def main():
    stockFile = open('kp2page.txt', 'r')

    htmls = stockFile.read()
    r = re.compile("code=([\d\w]*) \" target=\"_blank\">(.+)</a>")
    rs = r.findall(htmls)

    for m in rs:
        print "%s : %s" % (m[0], m[1].strip())

main()
```

스크립트를 돌리면 수초 안에 우리가 원했던 정보가 추출된다. 필자가 얼마나 무식하게 작업하고 있었는지 알 수 있는 부분이다. 수동적으로 작업을 하기 이전에 자동화 할 수 없는지 생각해 보는 것은 매우 중요하다. 기계가 할 수 있는 일을 우리가 대신 하는 것은 비효율 적이기 때문이다.

주식 데이터

코드 정리 다음 작업은 개별 페이지에서 우리가 원하는 정보를 가져오는 것이다. <화면 2>는 다음 증권에서 회사 주가를 조회하는 페이지다. 여기서 우리에게 필요한 정보는 빨간 글씨로 표시된 현재가, 전일비, 등락률이다.

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

화면 2 다음 페이지에서 주식 시세를 조회하는 화면

이 정보를 가공해서 <리스트 2>와 같은 XML 형태로 만든다. kospi와 kosdaq은 각각 코스피, 코스닥 지수를 나타낸다. stock은 사용자가 요청한 개별 주식에 대한 데이터가 들어간다. code는 주식 코드를, name은 회사명을, price는 현재가를, changedPrice는 전일비를, changedPercent는 등락률을, url은 해당 정보를 가지고 있는 다음 포털 페이지 주소를 나타낸다.

리스트 2 주가 정보를 표현한 XML

```
<?xml version="1.0" encoding="utf-8" ?>
<stocks>
<kospi>
    <price>1482.29</price>
    <changedPrice>18.54</changedPrice>
    <changedPercent>1.27%</changedPercent>
    <url>http://stock.finance.daum.net/trade/quote/total/inquiry/s3011000.html</url>
</kospi>
<kosdaq>
    <price>660.76</price>
    <changedPrice>5.54</changedPrice>
    <changedPercent>0.85%</changedPercent>
    <url>http://stock.finance.daum.net/trade/quote/total/inquiry/s3012000.html</url>
</kosdaq>
<stock>
    <code>005930</code>
    <name>삼성전자</name>
    <price>595,000</price>
    <changedPrice>21,000</changedPrice>
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
    <changedPercent>3.66%</changedPercent>
    <url>http://stock.finance.daum.net/search/quote/s2010000.html?code=005930</url>
</stock>
</stocks>
```

페이지에서 정보를 추출하는 방법은 앞서 살펴 보았던 코드명을 추출해 내는 것과 큰 차이가 없다. 페이지의 특정 부분을 정규 표현식을 사용해서 분리해 내면 된다. <리스트 3>에 나와있는 파이썬 스크립트는 다음 페이지를 분석해서 사용자가 요청한 주가 정보를 XML 형태로 제공하는 일을 한다.

리스트 3 페이지의 정보를 추출해서 XML 형태로 가공하는 파이썬 스크립트

```
#!/usr/local/bin/python
#-*- coding: euc-kr -*-

import re
import urllib
import cgi
import stockdb
import random

# 일반 회사 주가를 저장하고 있는 클래스
# 회사 code를 받아서 주식 정보를 분석해서 저장한다.
class Stock:
    def __init__(self, code):
        self.code = code;
        self.name = stockdb.Stocks[code]

        url = 'http://stock.finance.daum.net/search/quote/s2010000.html?code=' + code
        self.url = url
        htmls = urllib.urlopen(url).read()

        r = re.compile("""<dt>현재가</dt>\s*<dd class=".+">([\d,]*)</dd>\s*<dd class="""
            + """".+"><span>.*</span>([\d,]*)</dd>\s*<dd class=".+">([+-]*\d*\.\d*%)</dd>""")
        m = r.search(htmls)
        self.price = m.group(1)
        self.changedPrice = m.group(2)
        self.changedPercent = m.group(3)


    def printXML(self):
        print """<stock>
        <code>%s</code>
        <name>%s</name>
        <price>%s</price>
        <changedPrice>%s</changedPrice>
        <changedPercent>%s</changedPercent>
        <url>%s</url>
</stock>""" % (self.code, unicode(self.name, 'euc-kr').encode('utf8'), self.price
              , self.changedPrice, self.changedPercent, self.url)

# 특정 url의 주가를 저장해서 가지고 있는 클래스
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
# 주가 데이터를 포함하고 있는 url를 입력받아서 분석한다.
# kospi, kosdaq 주가를 표시하는데 사용한다.
class UrlStock:
    def __init__(self, tag, url):
        self.url = url;
        self.tag = tag;

        htmls = urllib.urlopen(url).read()

        r = re.compile("""현재가</b></td>\s*<td height="25" rowspan="3" align="right" """
            + """style="padding-right:8px">\s*<b><span style="font-size:15px; """
            + """color:#[\d\w]*">([\d.]*)""")
        m = r.search(htmls)
        self.price = m.group(1)

        r = re.compile("""전일비</b></td>\s*<td height="25"\s*align="right"\s*"""
            + """class="\w*"\s*style="padding-right:8px">\s*<b>((\s*<img """
            + """src=http://[\d\w/\.]*>\s*)|([\s▲▼]*))(\d*.\d*)""")
        m = r.search(htmls)
        self.changedPrice = m.group(4)

        r = re.compile("""등락률</b></td>\s*<td align="right" """
            + """class="\w*"\s*style="padding-right:8px"><b>\s*([+-]*\d*\.\d*%)"""
            + """\s*</b></td>""")
        m = r.search(htmls)
        self.changedPercent = m.group(1)


    def printXML(self):
        print """<%s>
        <price>%s</price>
        <changedPrice>%s</changedPrice>
        <changedPercent>%s</changedPercent>
        <url>%s</url>
</%s>""" % (self.tag, self.price, self.changedPrice, self.changedPercent, self.url, self.tag)

def main():
    print """<?xml version="1.0" encoding="utf-8" ?>"""
    print "<stocks>"

    # kospi, kosdap 주가를 먼저 표시 한다.
    kospiUrl = 'http://stock.finance.daum.net/trade/quote/total/inquiry/s3011000.html';
    UrlStock('kospi', kospiUrl).printXML();

    kosdaqUrl = 'http://stock.finance.daum.net/trade/quote/total/inquiry/s3012000.html';
    UrlStock('kosdaq', kosdaqUrl).printXML();

    # codes로 넘어온 주가를 분리해서 하나씩 표시 한다.
    form = cgi.FieldStorage()
    if form.has_key("codes") and form["codes"].value != "":
        codes = form["codes"].value.split()
        sa = set(codes)
        for code in sa:
            Stock(code).printXML()
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
    print "</stocks>"

print "Content-type: text/xml\n"
main()
```

이 스크립트는 http://www.jiniya.net/gadget/stock/sp.py에 설치되어 있다. 알고 싶은 회사 코드를 codes 변수에 넣어준다. 여러 회사에 대한 정보를 구하기 위해서는 구하고 싶은 code값들을 +로 연결해서 전달해주면 된다. http://www.jiniya.net/gadget/stock/sp.py?codes=005930+005935를 방문하면 삼성전자와 삼성전자우에 대한 주가를 담고 있는 XML 페이지를 볼 수 있다.

StockViewer 가젯

이제 StockViewer를 만들기 위한 준비 단계는 모두 끝났다. 앞서 어려운 일들을 모두 해결했기 때문에 실제 가젯 코드는 UI를 조작하는 코드와 XML 분석하는 코드가 전부다. 여기서는 간단하게 전역 변수와 사용된 데이터 구조의 역할에 대해서만 간략히 살펴본다. 실제 가젯의 세부적인 구현 사항은 이달의 디스켓을 참고하도록 하자.

<리스트 4>에 표시된 아홉 개의 전역 변수가 StockViewer의 핵심적인 구조를 보여준다. 네 개의 다이알 로그와 메뉴 바, 주식을 표시하는 테이블의 DOM 구조를 관리하는 클래스의 인스턴스가 변수에 저장된다. <리스트 5>에 나와있는 클래스는 실제 주식 데이터를 저장하기 위해서 사용된 것들이다. StockCode는 앞에서 살펴봤던 회사명과 주식코드를 저장하는 역할을 한다. <리스트 6>에 표시된 것과 같이 전역 변수 g_stockDB에 전체 회사명과 코드가 배열로 저장되어 있다. StockData는 사용자가 등록한 개별 회사에 대한 주가를 저장하는 용도로 사용된다. XML 파일의 내용이 파싱되어서 이 클래스에 저장된다.

리스트 4 StockViewer에 사용된 전역 변수

```
var g_stockViewer = null;         // 메인 프로그램 인스턴스
var g_searchDialog;                       // 검색 화면 인스턴스
var g_menuBar;                            // 메뉴 바 인스턴스
var g_stockTable;                         // 보기 화면 인스턴스
var g_aboutDialog;                        // 정보 화면 인스턴스
var g_initDialog;                         // 초기화 화면 인스턴스
var g_configDialog;                       // 설정 화면 인스턴스
var g_refreshRate = 30000;        // 갱신 주기
var g_searchNew = false; // 검색된 회사 검색하지 않음 옵션 값
```

리스트 5 주식 데이터를 나타내는데 사용된 데이터 구조

```
// KOSPI/KOSDAQ 기업 정보 데이터 객체
function StockCode(code, name)
{
    this.code = code;
    this.name = name;
}
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
// 실제 주가 종목 정보 데이터 객체
function StockData(code, name, price, changedPrice, changedPercent, url)
{
    this.code = code;
    this.name = name;
    this.price = price;
    this.changedPrice = changedPrice;
    this.changedPercent = changedPercent;
    this.url = url;
}
```

리스트 6 주가 검색을 위한 데이터

```
var g_stockDB = new Array(
    new StockCode("024810" , "이화전기"),
    new StockCode("040740" , "태원엔터테인"),
    new StockCode("048410" , "현대아이티"),
    new StockCode("025550" , "한국선재"),
    new StockCode("065060" , "지엔코"),
    new StockCode("066410" , "비트윈"),
    new StockCode("032190" , "다우데이타"),
    new StockCode("065570" , "삼영이엔씨"),
    new StockCode("006050" , "국영지앤엠"),
    new StockCode("054930" , "유신"),
    new StockCode("057030" , "YBM시사닷컴"),
    new StockCode("039060" , "인바이오넷"),
    new StockCode("026940" , "부국철강"),
    new StockCode("033260" , "쌈지"),
    new StockCode("042870" , "닛시"),
    new StockCode("052220" , "iMBC"),
```

XML 파싱

StockViewer는 주가 정보를 출력하기 위해서 sp.py에서 생성해준 XML 파일을 파싱해서 사용한다. <리스트 7>의 GetStockData는 sp.py에서 생성한 XML 파일을 읽어오는 일을 수행한다. OnStockDataRecv는 XML 파일 읽기 작업이 완료된 경우에 호출되고, 읽어온 XML 파일을 파싱해서 자바스크립트에서 사용할 수 있는 데이터 구조로 변경하는 일을 한다. HTML과 마찬가지로 XML 또한 DOM 계층 구조를 이루고 있기 때문에 첫 시간에 소개한 DOM 관련 함수들을 사용해서 접근하고 데이터를 읽어올 수 있다.

리스트 7 주가 XML 파싱 부분

```
// 주식 데이터를 읽어 온다.
this.GetStockData = function()
{
    // 타이머를 중지한다
    if(m_timerID)
    {
        clearTimeout(m_timerID);
        m_timerID = 0;
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
    }

    // 관심 종목을 구해온다
    var codes = g_stockViewer.GetConfigValue(m_codesKey);
    if(codes == null)
        return;

    // 애니메이션을 표시한다
    g_menuBar.ShowLoading(true);

    // url을 읽어온다
    var url = m_stockFeedUrl + codes + "&v=" + rand(100);
    var r = Web.Network.createRequest(
        Web.Network.Type.XML,
        url,
        {proxy:"generic"},
        OnStockDataRecv);

    r.execute();

    // 다음 번 읽기 작업을 위해 타이머를 예약한다
    m_timerID = setTimeout(g_stockViewer.GetStockData, g_refreshRate);
}

// 읽어온 XML을 분석하는 부분
function OnStockDataRecv(response)
{
    // 애니메이션을 중지 한다
    m_loadTimerID = setTimeout(HideLoading, 2000);

    // 읽기에 실패한 경우
    if(response.status != 200)
        return;

    var xml = response.responseXML.documentElement;
    if(xml)
    {
        // kospi 지수를 읽어 들인다
        var kospiNode = xml.getElementsByTagName('kospi');
        var kospi = new Object();
        kospi.price = kospiNode[0].getElementsByTagName('price').item(0).text;
        kospi.changedPrice = kospiNode[0].getElementsByTagName('changedPrice').item(0).text;
        kospi.changedPercent                                                    =
kospiNode[0].getElementsByTagName('changedPercent').item(0).text;
        kospi.url = kospiNode[0].getElementsByTagName('url').item(0).text;

        // kosdaq 지수를 읽어 들인다
        var kosdaqNode = xml.getElementsByTagName('kosdaq');
        var kosdaq = new Object();
        kosdaq.price = kosdaqNode[0].getElementsByTagName('price').item(0).text;
        kosdaq.changedPrice = kosdaqNode[0].getElementsByTagName('changedPrice').item(0).text;
        kosdaq.changedPercent
            = kosdaqNode[0].getElementsByTagName('changedPercent').item(0).text;
        kosdaq.url = kosdaqNode[0].getElementsByTagName('url').item(0).text;
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
        // 개별 종목에 대한 주가를 읽어 들인다
        var code;
        var stocks = xml.getElementsByTagName('stock');
        for (var i=0; i<stocks.length; ++i)
        {
            code = stocks[i].getElementsByTagName('code').item(0).text;

            m_stockData[code] = new StockData(  code,
                                stocks[i].getElementsByTagName('name').item(0).text,
                                stocks[i].getElementsByTagName('price').item(0).text,
                                stocks[i].getElementsByTagName('changedPrice').item(0).text,
                                stocks[i].getElementsByTagName('changedPercent').item(0).text,
                                stocks[i].getElementsByTagName('url').item(0).text);

        }

        // 주가 테이블을 갱신한다
        g_stockTable.Render(kospi, kosdaq, m_stockData);
    }
}
```

이벤트 매니저

Internet Explorer와 FireFox에서 이벤트를 처리하는 방법이 다르다. Internet Explorer는 attachEvent/detachEvent 함수를 사용해서 이벤트를 동록하고 해제한다. 반면에 FireFox에서는 addEventListener/removeEventListener를 사용해서 등록하고 해제한다. 따라서 브라우저에 따라서 이를 적절하게 처리해 주어야 한다.

이벤트 매니저(<리스트 8> 참고)는 브라우저에 따라 틀린 이벤트 등록 함수를 적절하게 처리해 주는 기능과 더불어서 등록된 이벤트를 한 번에 모두 제거하는 기능을 가지고 있다. 여러 군데에서 이벤트를 등록하는 경우에는 해제를 잊기 쉽기 때문에 일괄적으로 관리하는 것이 효과적이다.

한 가지 눈 여겨 볼 점은 브라우저의 종류를 판별하는 코드가 없다는 점이다. 브라우저의 종류에 따라서 함수를 나누면 알려지지 않은 브라우저에 대해서는 어떤 방법을 적용해야 하는지 결정할 수 없다. 반면에 <리스트 8>와 같이 함수의 지원 여부를 판별해서 사용하면 두 가지 중 하나라도 지원하는 브라우저에서는 모두 정상 동작한다.

리스트 8 이벤트 매니저 코드

```
// 이벤트 아이템
// obj: 이벤트를 동록한 오브젝트(DOM 객체)
// evt: 이벤트 종류
// fn: 이벤트 함수
function EventItem(obj, evt, fn)
{
    this.obj = obj;
    this.evt = evt;
```


이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
    this.fn = fn;
}

function EventManager()
{
    this.m_evtList = new Array();
}

// 등록된 모든 이벤트를 제거한다.
EventManager.prototype.Dispose = function()
{
    for(var i=0; i<this.m_evtList.length; ++i)
    {
        this.RemoveEvent(this.m_evtList[i].obj, this.m_evtList[i].evt, this.m_evtList[i].fn);
    }

    this.m_evtList = null;
}

// 이벤트를 추가한다.
EventManager.prototype.Add = function(obj, evt, fn)
{
    this.AddEvent(obj, evt, fn);
    this.m_evtList[this.m_evtList.length] = new EventItem(obj, evt, fn);
}

// 이벤트를 제거한다.
EventManager.prototype.Remove = function(obj, evt)
{
    for(var i=0; i<this.m_evtList.length; ++i)
    {
        if(this.m_evtList[i].obj == obj && this.m_evtList[i].evt == evt)
        {
            this.RemoveEvent(obj, evt, this.m_evtList[i].fn);
        }
    }
}

// 모든 이벤트를 제거하고 이벤트 리스트를 초기화 한다.
EventManager.prototype.RemoveAll = function()
{
    for(var i=0; i<this.m_evtList.length; ++i)
    {
        this.RemoveEvent(this.m_evtList[i].obj, this.m_evtList[i].evt, this.m_evtList[i].fn);
    }

    this.m_evtList = null;
    this.m_evtList = new Array();
}

// 실제로 이벤트를 추가하는 함수
EventManager.prototype.AddEvent = function(obj, evt, fn)
{
    if(obj.addEventListener)
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
        obj.addEventListener(evt, fn, false);
    else if(obj.attachEvent)
    {
        obj.attachEvent("on" + evt, fn);
    }
}

// 실제로 이벤트를 해제하는 함수
EventManager.prototype.RemoveEvent = function(obj, evt, fn)
{
    if(obj.removeEventListener)
        obj.removeEventListener(evt, fn, false);
    else if(obj.atachEvent)
        obj.detachEvent("on" + evt, fn);
}
```

이벤트 함수로 매개 변수 전달하기

자바스크립트에서 이벤트 함수가 호출되는 컨텍스트는 가변적이다. 이 말은 이벤트 함수를 등록할 때의 this와 해당 이벤트 함수가 호출 되었을 때의 this가 다르다는 말이다. 버튼 클릭 이벤트에 A라는 클래스의 onclick 함수를 등록했다고 생각해 보자. 이 상황에서 등록할 당시의 해당 이벤트 함수의 this는 A 클래스 인스턴스가 된다. 하지만 이벤트가 발생해서 함수가 호출될 때의 this는 버튼의 DOM 오브젝트다. 따라서 this를 사용해서 이벤트 함수로 정보 전들을 하는 방법은 사용할 수 없다.

이벤트 함수로 매개 변수를 전달하는 방법에는 크게 세 가지 정도의 방법이 있다. 가젯에서 사용되는 정석은 바인딩을 사용하는 것이다. 이제껏 우리가 만든 가젯의 메인 클래스도 하나의 바인딩이라 볼 수 있다. 이 방법에 대한 자세한 설명은 "가젯 개발자 가이드"에 자세히 나와있다. 나머지 두 가지는 클로저와 DOM 객체를 사용하는 방법이다.

클로저를 사용한 방법은 중첩 함수를 사용해서 해당 이벤트 핸들러 외부의 정보를 참조하는 방법이다(<리스트 9> 참고). 이 벙법은 간단하고 직관적이라는 장점이 있는 반면에 해당 함수가 수행할 때 마다 매번 새로운 이벤트 함수가 생성되는 것이기 때문에 불필요한 메모리를 많이 차지한다는 단점이 있다.

리스트 9 클로저를 사용해 이벤트 함수로 정보를 전달하는 방법

```
function registerEvent()
{
    var param = '전달할 정보';

    function onClick()
    {
        alert(param);
    }

    button.attachEvent('onclick', onClick);
```

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

```
}
```

DOM 객체를 사용하는 방법은 이벤트 핸들러로 전달할 정보를 DOM 객체에 미리 설정해 두는 방법이다(<리스트 10> 참고). 이 방법은 필자가 참고한 "Ajax 인 액션"이란 책에서 소개된 방법이다. 이 방은 무척 괜찮게 보이지만 필자가 테스트 해 본 결과 Internet Explorer와 FireFox의 이벤트 함수의 this로 설정되는 DOM 객체가 다르다는 단점이 있었다. 따라서 브라우저 특성을 타기 때문에 실제로 제대로 적용하기는 까다로운 방법이다.

리스트 10 DOM 객체를 사용해서 이벤트 함수로 정보를 전달하는 방법

```
function onClick()
{
    alert(this.param);
}

function registerEvent()
{
    button.param = '전달할 정보';
    button.attachEvent('onclick', onClick);
}
```

StockViewer는 전역 변수를 사용해서 이벤트 함수로 정보를 전달했다(<리스트 11> 참고). 거의 모든 객체가 전역으로 노출되어 있기 때문에 이벤트 함수에서는 그 객체들을 통해서 원하는 정보를 다루고 설정할 수 있다. 가장 무식하지만 가장 쉬운 방법이다. StockViewer의 소스가 길지 않기 때문에 가독성이나 유지보수에도 큰 영향을 미치지 않는다.

리스트 11 전역 변수를 사용해서 정보를 전달하는 방법

```
function OnClickSearchButton()
{
    var codes = g_stockViewer.GetCodes();
    var m_this = g_searchDialog;
    m_this.DeleteAllResults();

    // 중략

}
```

메모리 릭 점검하기

안그래도 느린 가젯은 메모리 릭에 의해 더욱 느려진다. 따라서 자신의 가젯에 메모리 릭이 있는지 없는지 점검하는 것은 매우 중요하다. "http://gadgets.start.com/gadget.aspx?manifestUrl=가젯 매니페스트파일 URL"에 접속하면 자신의 가젯에 릭이 있는지 없는지 간접적으로 체크할 수 있다.

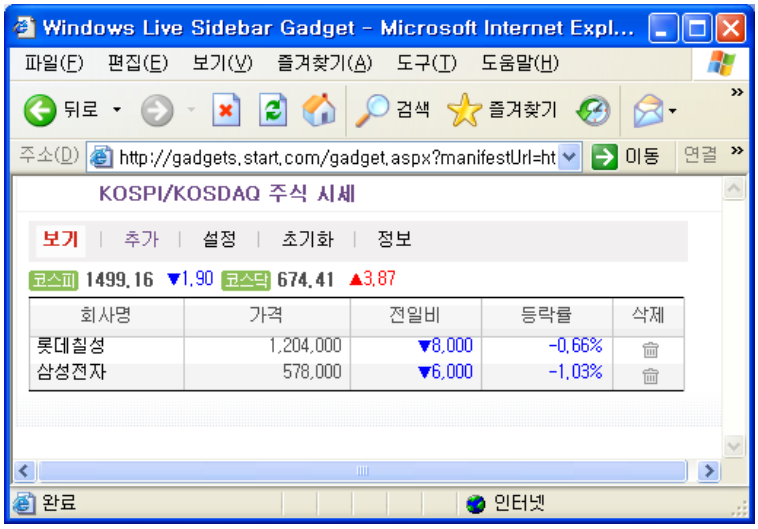
브라우저를 띄운 다음 해당 페이지를 열어 보자. 아마 자신의 가젯만 설치된 페이지가 뜰 것이다. <화면 3>은 StockViewer의 메모리 릭을 점검하는 화면이다. 이 상태에서 작업 관리자와 같은 프

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

로세스 모니터링 도구로 자신의 가젯이 떠있는 브라우저의 메모리 점유율을 체크한다. 가젯을 조작하는 과정에서 점유율이 늘어난다면 릭이 있는 것이다.

물론 이 방법을 100% 신뢰할 순 없다. 브라우저 자체적으로 메모리 릭이 발생할 수도 있고 캐쉬를 위해서 메모리가 바로 해제되지 않을 수도 있기 때문이다. 이 페이지로 점검하기 전에 코드상에서 dispose 메소드에서 자신이 생성한 DOM 오브젝트나 이벤트 메소드가 모두 해제되었는지를 꼼꼼히 살펴보는 것이 중요하다.



화면 3 StockViewer 가젯의 메모리 릭을 점검하는 화면

도전 과제

최근에 필자는 StockViewer에 주식 트래커 기능이 있었으면 좋겠다는 메일을 받았다. 주식 트래커란 투자자가 주식을 매입한 가격과 현재가를 비교해서 이익이 났는지 아니면 손해를 봤는지를 실시간으로 알려주는 것을 말한다. 주식 투자를 하는 투자자 입장에서는 현재가나 등락률 같은 내용보다는 자신의 매입가에서의 손익여부에 더 관심이 많기 때문에 투자자들에게 매력적인 기능이라 할 수 있다.

필자도 처음에 주식 트래커 기능을 추가하려고 했으나 개발 시간도 충분하지 않았고 출품을 위해서 마무리도 해야 했기에 추가하지는 못했다. 가젯 플랫폼에 관심 있는 독자라면 해당 기능이나 별도의 가젯으로 주식 트래커를 만들어 보는 것도 좋을 것 같다.

참고 자료

1 가젯 개발자 가이드

- http://218.38.34.168/gadget/Developer_Guide.doc

2 가젯 포럼

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/

http://microsoftgadgets.com/forums/6/ShowForum.aspx

3 가젯 API

http://microsoftgadgets.com/livesdk/docs/apiref.htm

4 가젯 갤러리

> http://microsoftgadgets.com/Gallery/

5 Ajax 인 액션, Dave Crane/Eric Pascarello/Darren James 저, ㈜에이콘 출판사

Copyright © 2007, 신영진

이 문서는 Creative Commons 라이센스를 따릅니다.
http://creativecommons.org/licenses/by-nc-nd/2.0/kr/