나만의 윈도우 라이브 가젯 만들기
Hello, World 가젯 만들기

목차

소개

우리는 그 동안 가젯에 사용될 자바스크립트에 대해서 충분히 익혔다. 이번 시간에는 live.com에 설치해서 사용할 수 있는 간단한 Hello, World 가젯을 만들고 live.com에 설치해 본다. 이를 위해 가젯을 구성하는 기본적인 세 가지 파일(매니페스트, 스크립트, 스타일 시트)과 핵심적인 가젯 API에 관해서 살펴본다.

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
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 시삽을 맡고 있으며, Microsoft Visual C++ MVP로 활동하고 있다. 최근에는 python과 lua같은 스크립트 언어를 배우려고 노력하고 있다.

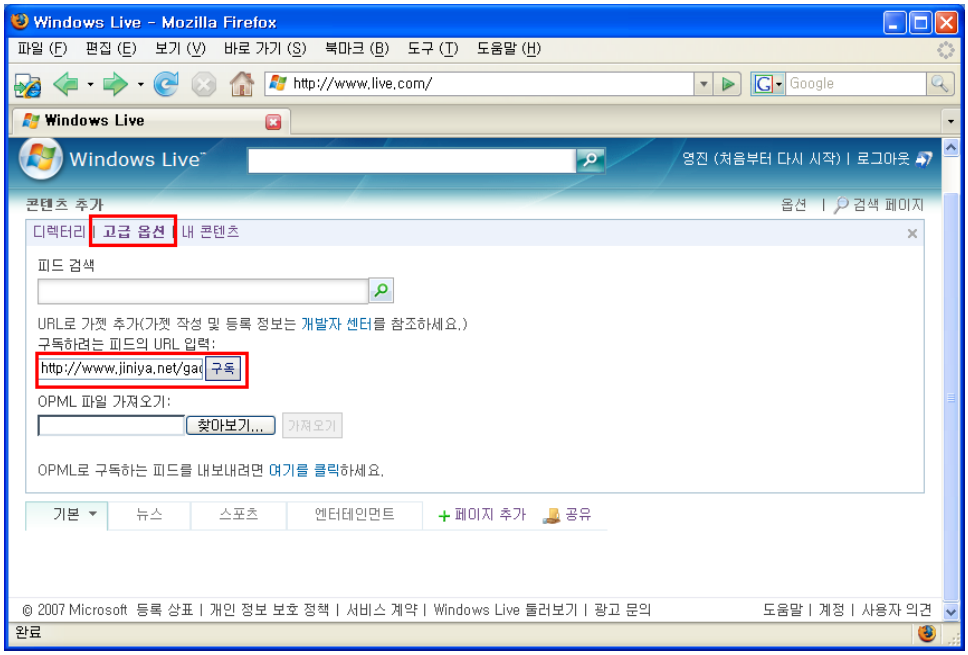
필자 메모

필자는 지난 3월 미국 시애틀에서 열리는 2007 MVP Global Summit에 참가했다. 유명하신 국내외 개발자 분들과 마이크로소프트 관계자 분들을 만나서 대화할 수 있는 아주 뜻 깊은 자리였다. 아이러니하게도 이 뜻 깊은 자리에서 필자가 가장 크게 느낀 부분은 영어 공부의 중요성이었다.

C언어를 아는 개발자는 전세계의 개발자들과 C언어로 이야기할 수 있다. C#을 아는 개발자는 마찬가지로 C#을 아는 개발자들과 이야기할 수 있다. 하지만 영어를 하는 개발자는 전세계의 어떤 개발자하고도 이야기할 수 있다. 영어는 글로벌 개발자로 가는 길이자 자신이 가진 지식의 진입 장벽을 낮추는 길이다.

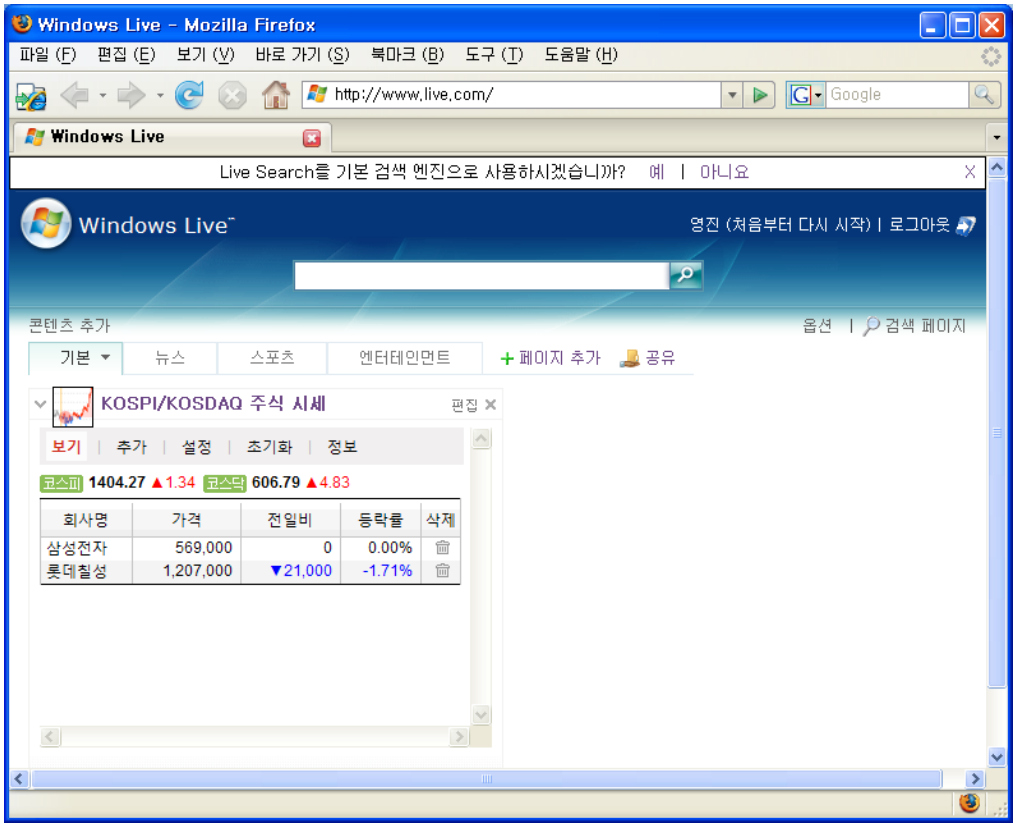
Introduction

지피지기면 백전백승이란 말이 있듯이 좋은 가젯을 만들기 위해서는 사전에 어떤 종류의 가젯들이 있는지, live.com에 어떻게 설치해서 사용하는지 알아보는 것이 우선이다. 가젯을 사용하려면 live.com에 접속해서 계정을 만들어야 한다. 로그인 한 다음 콘텐츠 추가 버튼을 누르고 고급 옵션을 선택한다(<화면 1> 참고). 해당 화면에서 구독하려는 피드 항목에 가젯의 매니페스트 XML 주소를 넣고 구독 버튼을 클릭하면 설치가 된다.



화면 1 live.com에서 가젯을 추가하는 화면

피드 URL에 http://www.jiniya.net/gadget/Stock/StockViewerGadget.xml을 입력해 보자. 입력한 다음 구독 버튼을 누르면 설치할지 물어보는 화면이 나타난다. 거기서 설치를 누르면 가젯을 사용할 수 있는 상태가 된다(<화면 2> 참고). 가젯을 조작해보고 어떤 식으로 동작하는지 살펴보도록 하자. "가젯 갤러리"를 방문하면 더 많은 가젯을 설치하고 사용해 볼 수 있다. 여러 종류의 가젯을 설치하고 사용해 보자.



화면 2 live.com에 StockViewer를 설치한 화면

가젯을 만드는데 꼭 필요한 삼총사

가젯은 기본적으로 세 개의 파일로 이루어 진다. 매니페스트 XML 파일, CSS 파일, 가젯의 핵심인 자바스크립트 파일이 그것이다. 매니페스트 파일은 가젯을 설치하는데 필요한 정보를 담고 있다. 어떠한 가젯인지, 어떤 스크립트 파일에 코드가 담겨 있는지 등의 정보가 저장되어 있다. CSS 파일은 가젯을 화면에 어떻게 나타낼지를 저장하고 있는 파일이다. 물론 이 파일 없이 스크립트 코드에서 일일이 스타일을 지정해 주어도 된다. 하지만 추후 유지보수가 힘들기 때문에 스타일은 별도의 CSS 파일로 분리하는 것이 좋다. 자바스크립트 파일은 가젯의 핵심적인 로직을 기록하고 있는 파일이다. 스크립트 파일은 지난 시간까지 배웠던 기본적인 자바 스크립트 지식과 함께 이번 시간에 소개할 가젯 API를 사용해서 만들어진다.

박스 1 웹 서버

가젯을 설치해서 사용하기 위해서는 웹 서버가 필요하다. 개인용 웹 서버에 설치해서 테스트하는 방법도 있지만 실제로 서비스 하기 위해서는 웹 서버를 구해서 설치해 보는 것이 좋다. 요즘은 웹 호스팅 가격이 저렴하기 때문에 손쉽게 리눅스 서버를 호스팅해서 사용 할 수 있다. 호스팅을 받기가 부담스럽다면 각자 PC에 IIS 서버를 설치해서 연습하면 된다. IIS 서버 설정 방법은 "가젯 개발자 가이드"에 자세히 나와 있다.

CSS 파일과 자바스크립트 파일은 일반적인 구조와 동일하기 때문에 매니페스트 파일의 구조에 대해서만 살펴보도록 하자. <리스트 1>에 전형적인 매니페스트 파일의 구조가 나와있다.

리스트 1 매니페스트파일 샘플

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:binding="http://www.live.com">
  <channel>
    // 가젯의 이름을 기록한다.
    <title>KOSPI/KOSDAQ 주식 시세</title>

    // 이름을 클릭했을 때 이동할 주소를 기록한다.
    <link>http://www.jiniya.net</link>

    // 가젯의 간단한 설명을 기록한다.
    <description>Korean Stock Viewer Gadget.</description>

    // 로케일을 지정한다.
    <language>ko-KR</language>

    // 가젯의 메인 클래스 이름을 적어준다.
    <binding:type>PNU.CSE.Stock.StockViewerGadget</binding:type>
    <item>
       // 가젯의 메인 스크립트 파일 이름을 적어준다.
       <link>http://www.jiniya.net/gadget/Stock/StockViewerGadget.js</link>
    </item>
    <item>
       // CSS 파일 이름을 지정해 준다.
       <link
binding:type="css">http://www.jiniya.net/gadget/Stock/StockViewerGadget.css</link>
    </item>
    <icons>
       // 가젯의 왼쪽 상단에 표시될 아이콘의 이름과 주소를 적어준다.
       <icon height="32" width="32">
              http://www.jiniya.net/gadget/Stock/StockViewerGadget.gif
       </icon>
    </icons>
  </channel>
</rss>
```

매니페스트 파일을 작성할 때 language 부분에 주의해야 한다. 이 부분은 인코딩을 적어주는 곳이 아니라 기본 로케일을 지정하는 곳이다. 인코딩은 문자를 저장하는 방식이고(utf8, euc-kr, latin1), 로케일은 언어와 지역을 나타내는 코드이다(en-US, ko-KR). 로케일에 대한 보다 자세한 내용은 "로케일 식별자와 문자열"에 나와있다.

가젯의 기본 구조

C 프로그램이 main에서 시작하는 것처럼 가젯도 특별한 규칙에 따라서 작성된 스크립트 파일만 live.com에 설치되어 사용할 수 있다. <리스트 2>에는 이러한 규칙을 준수하는 가장 단순한 형태

의 가젯 스크립트 구조가 나와있다.

리스트 2 더미 가젯 소스

```
// 가젯 네임스페이스를 등록한다.
registerNamespace("Microsoft.Live.GadgetSDK");

// 가젯의 생성자를 정의한다. 매니페스트에서 설정한 것과 일치해야 한다.
Microsoft.Live.GadgetSDK.HelloWorldGadget = function(p_elSource, p_args, p_namespace)
{
    // 항상 initializeBase를 가장 먼저 호출해야 한다.
    Microsoft.Live.GadgetSDK.HelloWorldGadget.initializeBase(this, arguments);

    this.initialize = function(p_objScope)
    {
        // 부모 클래스의 initialize 메소드를 호출한다.
        Microsoft.Live.GadgetSDK.HelloWorldGadget.getBaseMethod(this
            , "initialize", "Web.Bindings.Base").call(this, p_objScope);
    }
    Microsoft.Live.GadgetSDK.HelloWorldGadget.registerBaseMethod(this, "initialize");

    this.dispose = function(p_blnUnload)
    {
        // 부모 클래스의 dispose를 최종적으로 호출한다.
        Microsoft.Live.GadgetSDK.HelloWorldGadget.getBaseMethod(this
            , "dispose", "Web.Bindings.Base").call(this, p_blnUnload);
    }
    Microsoft.Live.GadgetSDK.HelloWorldGadget.registerBaseMethod(this, "dispose");
}
Microsoft.Live.GadgetSDK.HelloWorldGadget.registerClass("Microsoft.Live.GadgetSDK.HelloWorldG
adget", "Web.Bindings.Base");
```

처음 가젯 소스를 보는 분들이라면 굉장히 당황스러울 것이다. 아마 registerNamespace, registerBaseMethod 등의 함수에 대한 궁금증이 무척이나 클 것 같다. 하지만 안타까운 점은 그러한 궁금증에 대해서 필자가 명확하게 답변해 줄 수 없는 상태라는 점이다. 아직까지 라이브 가젯은 베타 상태이고 구조에 대한 명확한 문서가 없다. 그저 대부분의 개발자가 샘플을 보고 주먹구구 식으로 끼워 맞추기 형태로 만들고 있는 실정이다.

registerNamespace는 네임 스페이스를 등록하는 기능을 한다. registerNamespace가 없다면 다음 줄은 성립될 수가 없다. Microsoft 객체가 없다는 에러를 낼 것이다. registerNamespace는 Microsoft.Live.GadgetSDK가 오브젝트인 것처럼 만들어 주는 역할을 한다.

```
Microsoft.Live.GadgetSDK.HelloWorldGadget = function(p_elSource, p_args, p_namespace)
```

다음으로 나오는 것은 가젯의 생성자다. 총 세 개의 파라미터가 넘어온다. p_elSource는 가젯을 담고 있는 div 콘테이너 오브젝트가 넘어온다. 출력할 DOM 오브젝트를 생성한 다음 p_elSource에 추가해주면 해당 오브젝트들이 화면에 출력된다. p_args에는 각종 정보가 넘어온다. <표 1>에

는 Hello, World 가젯의 p_args로 넘어온 정보를 출력한 내용이 나와있다. 끝으로 p_namespace는 이름상으로는 네임스페이스 정보가 넘어오는 듯 하나, 실제로는 항상 null 값이 넘어왔다.

표 1 p_args로 넘어오는 정보들

멤버명	타입	값
feed	오브젝트	[object Object]
feedUrl	string	http://jiniya.net/gadget/Hello/HelloWorldGadget.xml
xml	object	[object XMLHttpRequest]
defaults	object	[object Object]
legacyscript	string	http://stj.live.com/live/1.900.8.030/js/legacy.js
mode	string	author
view	string	Custom
hostmode		null
id	string	1933d5f8-8b47-4279-bf96-8660c3a5ff74
fullMode	bool	false
mkt	string	ko-kr
loc	string	kr
lang	string	ko
params	object	[object Object]
moduleDef		null
isolated	int	0
module	object	[object Object]
gadgetHost	object	[object Object]
uri	string	http://jiniya.net/gadget/Hello/HelloWorldGadget.xml
listIcon		null
__lsid	string	10d51b3e-3274-47e2-9375-3f9bc8ae6b8e10d51b3e-3274-47e2-9375-3f9bc8ae6b8e
onDashboard	bool	true
xmlSources	string	빈 문자열

박스 2 오브젝트 내의 멤버 값 출력 하기

자바스크립트는 객체의 멤버를 사전과 같은 형태로 취급한다. 따라서 아래와 같은 간단한 함수를 통해서 특정 객체의 모든 멤버의 이름과 값을 출력해 볼 수 있다.

```
function BrowseObject(obj)
{
   var k;
   var str = '';
```

```
    for(k in obj)
    {
        str += k + '=>' + obj[k] + '\n';
    }

    return str;
}
```

생성자 내부로 들어가면 가장 먼저 만나는 것은 initializeBase를 호출하는 일이다. 이 함수는 부모 클래스를 초기화하는 일을 담당한다. this는 현재 컨텍스트 정보다. arguments는 생성자로 넘어온 세 개의 인자를 담고 있는 배열이다.

박스 3 arguments 사용 예

자바스크립트는 함수 호출에 대해서 인자의 타입이나 개수를 체크하지 않는다. 인자 세 개를 받아 들이는 foo라는 함수를 생각해보자. foo()와 같이 호출하면 인자 세 개는 모두 undefined로 넘어온다. foo(1)과 같이 호출하면 첫 번째 인자는 1이 되고, 나머지 두 개는 undefined가 된다. 물론 인자가 세 개인 foo에 네 개의 인자를 전달해서 호출하는 것도 가능하다.

자바스크립트의 모든 함수는 기본적으로 가변인자를 사용한다. 이러한 이유로 함수 시그니처에 포함된 인자 목록이 아닌 실제로 전달된 인자 목록에 접근할 변수가 필요하다. 함수 내의 arguments라는 내장 변수가 그러한 일을 한다. 이 변수는 배열의 형태로 구성되어 있고, 함수로 전달된 인자 목록을 내용으로 가진다. 이러한 arguments의 역할을 가장 극명하게 보여주는 것이 아래 나와있는 sprintf 함수의 자바스크립트 구현이다.

```
function sprintf()
{
    if (!arguments || arguments.length < 1 || !RegExp)
        return;

    var str = arguments[0];
    var re = /([^%]*)%('.|0|\x20)?(-)?(\d+)?(\.\d+)?(%|b|c|d|u|f|o|s|x|X)(.*)/;
    var a = b = [], numSubstitutions = 0, numMatches = 0;
    while (a = re.exec(str))
    {
        var leftpart = a[1], pPad = a[2], pJustify = a[3], pMinLength = a[4];
        var pPrecision = a[5], pType = a[6], rightPart = a[7];

        numMatches++;
        if (pType == '%')
        {
            subst = '%';
        }
        else
        {
            numSubstitutions++;
```

```
            if (numSubstitutions >= arguments.length)
            {
                alert('인자 개수가 부족 합니다!!!');
                return str;
            }

            var param = arguments[numSubstitutions];
            var pad = '';

            if (pPad && pPad.substr(0,1) == "'") pad = leftpart.substr(1,1);
            else if (pPad) pad = pPad;

            var justifyRight = true;
            if (pJustify && pJustify === "-") justifyRight = false;

            var minLength = -1;
            if (pMinLength) minLength = parseInt(pMinLength);

            var precision = -1;
            if (pPrecision && pType == 'f') precision = parseInt(pPrecision.substring(1));

            var subst = param;
            if (pType == 'b') subst = parseInt(param).toString(2);
            else if (pType == 'c') subst = String.fromCharCode(parseInt(param));
            else if (pType == 'd') subst = parseInt(param) ? parseInt(param) : 0;
            else if (pType == 'u') subst = Math.abs(param);
            else if (pType == 'f')
            {
                var prec = Math.pow(10, precision);
                subst = (precision > -1)
                    ? Math.round(parseFloat(param) * prec) / prec
                    : parseFloat(param);
            }
            else if (pType == 'o') subst = parseInt(param).toString(8);
            else if (pType == 's') subst = param;
            else if (pType == 'x') subst = ('' + parseInt(param).toString(16)).toLowerCase();
            else if (pType == 'X') subst = ('' + parseInt(param).toString(16)).toUpperCase();
        }

        str = leftpart + subst + rightPart;
    }

    return str;
}
```

initialize와 dispose는 각각 초기화와 종료 시에 불리는 함수 들이다. 특히 dispose에서는 가젯에서 사용한 객체들을 모두 소거해서 메모리 누수가 잃어나지 않도록 하는 것이 중요하다. getBaseMethod는 부모 클래스의 메소드를 찾는 기능을 한다. registerBaseMethod는 다형성을 구현하기 위해서 사용되는 것으로 추측된다. C++에서 virtual 메소드를 선언하기 위한 것이라고 생각하면 된다. initialize와 dispose외에 사용자가 추가한 메소드는 registerBaseMethod를 해주지 않

아도 된다. 끝으로 마지막 줄에 있는 registerClass는 우리가 만든 클래스를 등록하는 기능을 한다. 두 번째 인자인 Web.Bindings.Base가 우리가 만든 가젯의 부모 클래스이다.

Hello, World 가젯

이제 가젯을 만들기 위한 기본적인 지식은 모두 익혔다. 가장 기본적인 Hello, World 가젯을 만들어 보도록 하자. <리스트 3>에는 가젯의 매니페스트 파일이, <리스트 4>에는 가젯 자바스크립트 파일이 나와있다. 단순히 화면에 Hello, World를 출력하는 일만 하기 때문에 별도로 CSS 파일을 만들진 않았다.

리스트 3 HelloWorldGadget.xml 파일

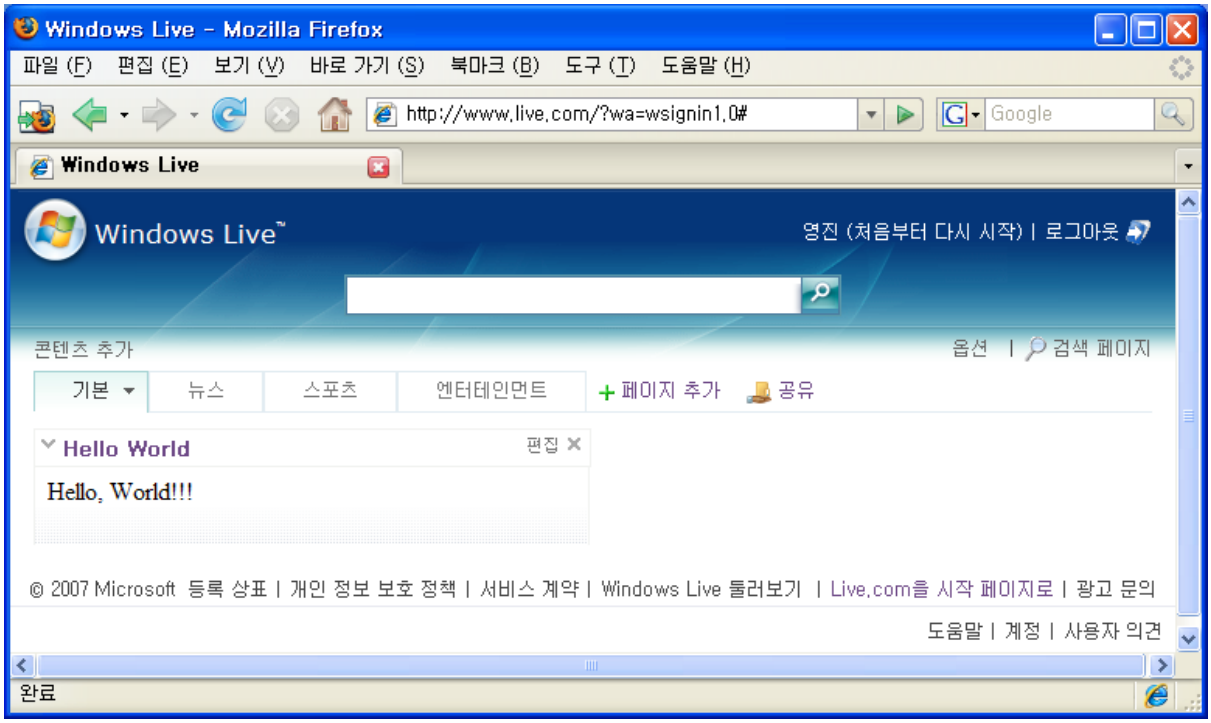
```
<?xml version="1.0"?>
<rss version="2.0" xmlns:binding="http://www.live.com">
  <channel>
    <title>Hello World</title>
    <link>http://www.jiniya.net</link>
    <description>A sample hello world binding.</description>
    <language>en-us</language>
    <binding:type>Microsoft.Live.GadgetSDK.HelloWorldGadget</binding:type>
    <item>
      <link>http://www.jiniya.net/gadget/Hello/HelloWorldGadget.js</link>
    </item>
  </channel>
</rss>
```

리스트 4 HelloWorldGadget.js

```
registerNamespace("Microsoft.Live.GadgetSDK");

Microsoft.Live.GadgetSDK.HelloWorldGadget = function(p_elSource, p_args, p_namespace)
{
    Microsoft.Live.GadgetSDK.HelloWorldGadget.initializeBase(this, arguments);

    this.initialize = function(p_objScope)
    {
        Microsoft.Live.GadgetSDK.HelloWorldGadget.getBaseMethod(this
            , "initialize", "Web.Bindings.Base").call(this, p_objScope);

        p_elSource.innerHTML = "Hello, World!!!";
    }
    Microsoft.Live.GadgetSDK.HelloWorldGadget.registerBaseMethod(this, "initialize");

    this.dispose = function(p_blnUnload)
    {
        Microsoft.Live.GadgetSDK.HelloWorldGadget.getBaseMethod(this
            , "dispose", "Web.Bindings.Base").call(this, p_blnUnload);
    }
    Microsoft.Live.GadgetSDK.HelloWorldGadget.registerBaseMethod(this, "dispose");
}
Microsoft.Live.GadgetSDK.HelloWorldGadget.registerClass(
```

```
    "Microsoft.Live.GadgetSDK.HelloWorldGadget", "Web.Bindings.Base");
```

이전에 살펴본 내용과 큰 차이가 없기 때문에 어려운 내용은 없다. 두 개의 파일을 작성한 다음 live.com에 설치해 보도록 하자. 자신의 웹 서버 주소에 맞게 링크 주소를 수정한 다음 live.com에 가서 구독해 보고 화면에 어떻게 출력되는지 살펴본다. 간단한 과정이지만 쉽지 않을 것이다. 필자도 처음 가젯을 만들 때 이 과정에서 타이핑 오류와 같은 사소한 오류로 꽤나 고생했다. 설치해서 <화면 3>과 같이 나타난다면 성공한 것이다.



화면 3 Hello, World 가젯 결과 화면

위 코드의 가장 핵심 적인 부분은 p_elSource에 Hello, World를 추가하는 것이다. 복잡한 가젯도 결국은 p_elSource에 출력할 DOM 오브젝트를 만들어서 추가하고 그것들의 이벤트를 받아서 동작하는 것이 전부다.

웹과 통신하는 방법

대다수 가젯은 자신이 직접 정보를 생성하기 보다는 다른 곳의 정보를 실시간으로 읽어와서 화면에 뿌려주는 역할을 한다. 따라서 다른 웹 페이지를 읽어와서 분석하는 일이 필수적이다. 이렇게 다른 웹에 존재하는 리소스를 읽어오는 함수로 Web.Network 클래스의 createRequest가 있다.

```
var netReq = Web.Network.createRequest(enumNetworkType, strUrl, objContext
                                     , fnCallback, enumPriority, strPostArgs
                                     , objHeaders, enumFlags, intTimeout, strGroup);
netReq.execute();
```

enumNetworkType – 요청할 대상의 리소스 타입을 지정한다. Web.Network.Type.CSS, Web.Network.Type.Image, Web.Network.Type.Script, Web.Network.Type.XML, Web.Network.Type.XMLGet 중에 한 가지 값을 가진다.
strUrl – 요청할 리소스의 URL을 넘겨준다.
objContext – 콜백 함수로 전달될 컨텍스트 오브젝트를 넣어준다. 추후에 콜백 함수에서 필요한 객체를 넘겨준다.
fnCallback – 리소스를 다운 받았을 때 호출 되는 콜백 함수를 지정한다. 지정하지 않으면 리소스가 다운로드 완료된 이벤트를 통지 받지 못한다.
enumPriority – 요청에 대한 우선순위를 지정한다. Web.Utility.Prioritizer.Priorities.High, Web.Utility.Prioritizer.Priorities.Medium, Web.Utility.Prioritizer.Priorities.Low, Web.Utility.Prioritizer.Priorities.Lowest 중에 한 가지를 지정할 수 있다. 지정하지 않으면 이미지는 낮은 우선 순위를 다른 리소스는 보통 수준의 우선 순위를 갖는다.
strPostArgs – xml 요청인 경우에 send 메소드로 전달될 postString을 지정한다. 요청 타입이 다른 경우엔 무시된다.
objHeaders – XMLPost 리소스에 대한 연관 배열이다.
enumFlags – 외부 요청이 어떻게 이루어질지 결정하는 플래그다. Web.Network.Flags.SERIALIZE, Web.Network.Flags.DUPLICATE를 조합해서 사용할 수 있다.
intTimeout – 요청에 대한 타임 아웃 시간을 지정한다. 지정하지 않으면 브라우저의 기본 설정에 따라 처리된다.
strGroup – 이 인자를 통해서 개별적인 요청을 그룹별로 관리할 수 있다. Web.Network.AbortGroup(strGroup) 함수를 통해서 특정 그룹에 속한 모든 요청을 취소할 수 있다.

createRequest 함수가 굉장히 복잡해 보이지만 일반적으론 앞의 네 가지 인자만 사용한다. <리스트 5>에는 StockViewer에서 주가 데이터 XML을 읽어오는 부분이 나와있다. 콜백 함수와 url 정도만 직접 넣어주면 된다. createRequest는 성공한 경우에 요청 객체를 리턴 한다. 실제로 해당 리소스를 얻어오기 위해서는 요청 객체의 execute 메소드를 호출해 주어야 한다.

리스트 5 StockViewer의 주가 데이터 XML을 읽어오는 부분

```
this.GetStockData = function()
{
   if(m_timerID)
   {
      clearTimeout(m_timerID);
      m_timerID = 0;
   }

   var codes = g_stockViewer.GetConfigValue(m_codesKey);
   if(codes == null)
      return;
```

```
    g_menuBar.ShowLoading(true);

    // url을 생성한다.
    var url = m_stockFeedUrl + codes + "&v=" + rand(100);

    // 요청 객체를 생성한다.
    var r = Web.Network.createRequest(
        Web.Network.Type.XML,
        url,
        {proxy:"generic"},
        OnStockDataRecv);

    // XML 파일을 읽어 온다.
    r.execute();

    // 일정 시간 후에 다시 GetStockData 함수를 호출한다
    m_timerID = setTimeout(g_stockViewer.GetStockData, g_refreshRate);
}

// XML 파일 읽기 완료 콜백
function OnStockDataRecv(response)
{
    // 200번은 성공한 경우다.
    if(response.status == 200)
    {
        var xml = response.responseXML.documentElement;
        if(xml)
        {
            var kospiNode = xml.getElementsByTagName('kospi');
            var kospi = new Object();
            kospi.price = kospiNode[0].getElementsByTagName('price').item(0).text;

            // ... 중략 ...

            g_stockTable.Render(kospi, kosdaq, m_stockData);
        }
    }

    m_loadTimerID = setTimeout(HideLoading, 2000);
}
```

박스 4 url 캐시 문제 해결하기

Web.Network.createRequest로 정보를 읽어올 때 주의해야 할 점은 캐싱이다. 일반적인 경우에 캐싱은 효율성을 높여주지만 주기적으로 해당 내용을 참고해서 사용자에게 알려주는 가젯에서는 문제가 발생할 수 있다. 실제로 StockViewer 또한 사용자가 설정한 동일한 주소의 XML 데이터를 일정 시간 간격으로 뿌려 주어야 하기 때문에 문제가 있었다. 문제를 해결하는 가장 손쉬운 방법은 url에 더미 데이터를 추가 시켜서 캐싱되지 않도록 하는 것이다. 아래 코드와 같이 url 뒤에 랜덤 값을 추가 시켜서 캐싱을 방지하면 된다.

```
var url = m_stockFeedUrl + codes + "&v=" + rand(100);
```

환경 정보를 저장하는 방법

대다수 가젯이 사용자의 설정 정보를 바탕으로 동작한다. 아주 간단한 가젯의 경우에도 한두 가지의 설정 값은 가지고 있다. 가젯은 이러한 일에 사용하기 위한 세 가지 종류의 API를 제공한다. 환경 설정 정보를 쓰고, 읽고, 지우는 것이 그것이다. 아래 API들은 module 객체의 멤버다. module 객체는 가젯의 생성자로 넘어온 p_args의 module 멤버를 통해서 참조할 수 있다.

setPreference(key, value); // 정보를 기록한다.
getPreference(key); // 기록된 정보를 읽어 온다.
deletePreference(key); // 기록된 정보를 삭제한다.

기본 적인 구조는 INI 파일과 유사하다. 각 가젯은 문자열 key에 해당하는 value(정보)를 저장할 수 있다. 해당 key 값을 통해서 정보를 읽고, 지울 수 있다. value 값으로는 임의의 자바스크립트 객체를 사용할 수 있다. key 값이 존재하지 않는 경우에 getPreference는 null을 리턴 하고, deletePreference는 아무런 작업도 하지 않는다.

가젯은 환경 설정 정보를 저장하기 위해서 1000문자 정도의 공간을 할당하고 있다. 1000문자에는 value의 값뿐만 아니라 key값과 부가적인 정보 저장 공간 까지 포함되기 때문에 대량의 데이터를 저장하는 것을 피하는 것이 좋다.

<리스트 6>에는 StockViewer에 사용된 환경설정 관련 코드가 나와있다. m_module은 생성자로 넘어온 p_args의 module 멤버를 저장해둔 것이다.

리스트 6 StockViewer에 사용된 환경 설정 부분

```
this.GetConfigValue = function(key)
{
    var value = m_module.getPreference(key);
    if(value == null || value == undefined || value == "")
        return null;

    return value;
}

this.SetConfigValue = function(key, val)
{
    m_module.setPreference(key, val);
}
```

module 클래스에는 이러한 환경 정보를 저장하는 기능 외에도 유용한 함수들을 다수 포함하고 있다. <표 2>에 이러한 함수들의 목록이 나와 있다. 개별 함수들의 자세한 사용법은 "가젯 API"를

참고한다.

표 2 module 클래스에 포함된 멤버 함수들

메소드	설명
getEI	가젯을 포함하고 있는 DOM 노드 객체를 구한다.
getFooterEI	가젯의 하단에 출력되는 DOM 노드 객체를 구한다(인라인 가젯에만 적용).
getId	가젯의 엘리먼트 ID를 구한다.
getLanguage	현재 사용되고 있는 언어를 구한다.
getLink	타이틀을 클릭했을 때 이동하는 가젯의 URL을 구한다.
getLocale	현재 사용되고 있는 로케일을 구한다.
getMarket	현재 사용하고 있는 언어, 로케일 문자열(en-US, ko-KR)을 반환한다.
getMode	가젯의 모드를 반환한다. author이나 viewer 중 하나다.
getTitle	가젯의 타이틀을 반환한다.
resize	가젯의 크기를 변경한다.
resolveUrl	매니페스트 URL을 기준으로 만든 절대 경로를 반환한다.
setFooterText	가젯의 하단 텍스트를 설정한다(인라인 가젯에만 적용).
setTitleIcon	가젯의 타이틀 바에 사용되는 아이콘을 변경한다(인라인 가젯에만 적용).
setTitleLink	타이틀 바를 클릭했을 때 이동할 URL을 변경한다(인라인 가젯에만 적용).
setTitleText	가젯의 타이틀 바의 글자를 변경한다(인라인 가젯에만 적용).

인라인 가젯이란 마이크로소프트의 인증을 받은 가젯들로 기본적으로 설치되는 대부분의 가젯이 여기에 포함된다. 이 가젯들의 가장 큰 특징은 별도의 iframe에 설치되지 않고 메인 페이지의 div에 바로 설치된다는 점이다. 따라서 메인 페이지에 변경을 가할 수도 있고, 여러 가지 속성을 추가적으로 설정할 수 있다. 대부분의 서드파티 업체에서 제작하는 가젯은 일반 가젯이다. 이 가젯들은 별도의 iframe에 불려지고 몇 가지 특수한 속성을 변경할 수 없도록 제한 받는다.

박스 5 임베딩 언어

소프트웨어란 말의 핵심은 소프트란 단어다. 고치기 쉽다는 말이다. 하지만 시대가 흐르고 CPU 속도가 올라갈수록 소프트의 기준도 변화하고 있다. 과거 CPU 파워가 부족하던 시대에는 서킷을 직접 수정하지 않는 수준의 소프트 함으로도 만족했다. 하지만 요즘 개발자는 그 정도론 만족하지 못한다. 그래서 최근에는 다이나믹 언어라고 불리는 스크립트 언어가 인기다.

하지만 아직은 스크립트 언어로 하기 힘든 일들이 많이 존재하고, 여전히 스크립트 언어는 느리기 때문에 둘을 섞어서 사용하는 방법이 많이 사용된다. 이렇게 컴파일러 언어에 붙여서 사용하

는 언어를 임베딩 언어라 한다. lua가 이러한 용도로 만들어진 대표적인 언어다.

lua는 훌륭한 임베딩 언어지만 지원하는 자료 구조나 표현식이 범용 스크립트 언어에 비해서 떨어진다는 단점이 있다. 또한 lua 문법을 새로 익혀야 한다는 점도 단점이라고 할 수 있다. 이러한 이유로 범용 스크립트 언어로 제작된 것들을 임베딩 시켜서 사용할 수 있도록 만드는 작업히 최근 활발하게 이루어지고 있다. 우리가 그 동안 열심히 익힌 자바스크립트도 모질라의 SpiderMonkey 엔진(http://www.mozilla.org/js/spidermonkey/)을 사용하면 C 프로그램에 임베딩 시켜서 사용할 수 있다.

도전 과제

이번 시간까지 배운 지식을 이용하면 어떠한 가젯이든 만들 수 있다. 이제 문제는 상상력이다. 필자가 제안해 볼 가젯은 지난 시간에 작성했던 체스 기보 뷰어의 연장선상에 있는 가젯이다. 체스를 하는 많은 사람들은 주기적으로 새로운 기보를 보기를 원한다. 이러한 기보 정보를 제공하는 웹 프로그램과 그 정보를 읽어와서 가젯에서 보여주도록 만들면 좋겠다. 단순히 기보만 보는 것으론 재미가 떨어지기 때문이 각 수가 진행될 때마다 풀이 내지는 메시지를 넣어서 어떤 상황인지 알려주도록 만들어 보자.

참고 자료

1 가젯 개발자 가이드

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

2 가젯 포럼

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 로케일 식별자와 문자열

> http://msdn2.microsoft.com/en-us/library/ms776260.aspx