
Hajo Schulz, Noud van Kruysbergen

# Voor op het prikbord

## Windows-gadgets voor op de desktop zelf ontwikkelen

**Sinds Windows Vista is het mogelijk zogenaamde gadgets, kleine programmaatjes die snel toegang bieden tot bepaalde informatie of social media, direct op de desktop vast te prikken. Voor het maken van gadgets kun je volstaan met een simpele teksteditor.**

Het kan handig zijn om bepaalde informatie tijdens je computeractiviteiten steeds in het oog te kunnen houden zonder eerst ergens op te moeten klikken. Neem bijvoorbeeld de datum, de tijd en het weer, of de laatste nieuwtjes van een (regionale) krant of voor technisch georiënteerde gebruikers wat de processorbelasting en het geheugengebruik is. Sinds Vista is het in Windows mogelijk dergelijke informatiebrokken direct op de desktop te zetten en automatisch bij te laten werken. De basis daarvoor zijn zogeheten desktopgadgets.

Microsoft had bij Windows Vista een smalle zone aan de rechterkant van het scherm gereserveerd voor die miniprogrammaatjes: de Sidebar. Sinds Windows 7 kun je gadgets op iedere willekeurige plek op de desktop zetten. Een kleine verzameling wordt bij Windows meegeleverd. Bij Windows Vista kom je daar via het kleine +-symbool aan de bovenkant van de Sidebar, bij Windows 7 kan dat met een rechtsklik op een leeg deel van de desktop en dan kiezen voor 'Gadgets'. In beide gevallen biedt het venster dat dan verschijnt rechts onderin een link naar 'Meer gadgets downloaden', waarmee je naar een apart deel van de website van Microsoft gaat, waar meer gadgets te downloaden zijn.

Het aanbod daar is van een zeer middelmatige kwaliteit, en de zoekfunctie levert bij de Nederlandse gadgets zelden een hit op. Toch hoef je niet bij de pakken neer te zitten, als een functie die je graag wilt hebben er niet bij zit. Het zelf maken van een desktopgadget is namelijk geen heksentoer en afhankelijk van de gewenste functionaliteiten redelijk snel zelf te programmeren met een overzienbare hoeveelheid werk.

### Handwerk

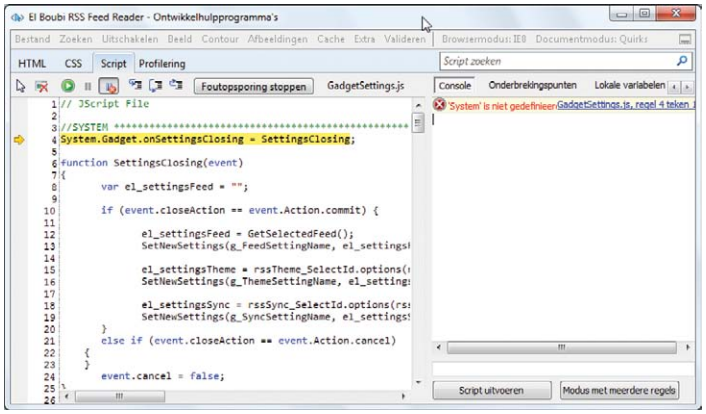
In principe is een gadget niets anders dan een kleine, lokaal opgeslagen webpagina. De inhoud wordt dan ook in HTML geschreven, waarbij de boel tot leven wordt gewekt met JavaScript-code. Om zoiets zelf te maken, heb je in essentie niet meer nodig dan een teksteditor als Notepad van Windows zelf. Om grafische elementen te maken, kun je terugvallen op Paint. Als je het iets makkelijker wilt doen, neem je een editor met syntax-highlighting en automatisch aanvullen voor HTML en JavaScript, bijvoorbeeld het opensource programma Notepad++ of de freeware PSPad. Alle genoemde tools zijn via de softlink aan het eind van dit artikel te vinden. Een WYSIWYG-HTML-editor heeft bij het maken van een gadget geen voordelen: de kleine beschikbare beeldschermruimte dwingt je tot een pixelprecieze lay-out, waarbij je vroeg of laat toch handmatig de HTML-code moet aanpassen. Voor het tekenen van afbeeldingen is een programma handig dat goed met transparantie kan omgaan. Ook laagfuncties zijn praktisch. Populaire freeware programma's die dat kunnen zijn bijvoorbeeld Paint.NET en GIMP.

Een heikel punt bij het programmeren van gadgets is het opsporen van fouten in de JavaScript-code. Als je Microsofts ontwikkelomgeving Visual Studio hebt, heb je het verhoudingsgewijs makkelijk. Deze IDE heeft aan de ene kant een bruikbare editor en aan de andere kant een JavaScript-debugger. Je kunt die echter niet zoals gebruikelijk starten met F5, maar je moet je gadget handmatig starten en dan in het proces sidebar.exe hangen. Dat laatste kan alleen met de commerciële versies van Visual Studio, waardoor de gratis Visual Web Developer Express als ontwikkelomgeving voor gadgets helaas afvalt.

Als je geen geld wilt uitgeven, dan heb je voor het debuggen altijd nog de Ontwikkelhulpprogramma's van Internet Explorer 8. Je laadt de HTML-pagina van de gadget, bevestigt dat het uitvoeren van scripts in dit geval onschadelijk is en drukt dan op F12 om de Ontwikkelhulpprogramma's op te roepen. Als je dan op het tabblad 'Script' onder het menu klikt, krijg je een werkbalk met de voor een debugger gebruikelijke knoppen voor het starten, onderbreken en stapsgewijs uitvoeren van de code. In de rechterhelft van het venster kun je dan de inhoud van variabelen en de onderbrekingspunten laten zien.

De debugger zal bij het opstarten meteen al met een paar fouten komen. De gadgetcode gebruikt het JavaScript-dialect van

Internet Explorer, want die is in principe verantwoordelijk voor het laten zien van de gadgets. Maar tijdens het uitvoeren van een gadget zijn er een paar objecten en functies die per se nodig zijn, maar die een normale Internet Explorer niet kent. Daar horen onder andere de opties bij die een gebruiker van de gadget kan instellen via het steeksleutelpictogram. Die worden uitgelezen met System.Gadget.Settings.read(), maar de IE-debugger komt dan met de melding dat 'System' niet gedefinieerd is. Als ontwikkelaar kun je in dit geval niets anders doen dan door deze fout heen klikken en de inhoud van de betreffende variabelen handmatig op de verwachte waarden zetten.




**Het debuggen van de gadgetcode gaat met de Ontwikkelhulpprogramma's van Internet Explorer wat moeizaam.**

## Aan de slag

Gadgets die door de gebruiker zelf zijn geïnstalleerd, staan op de harde schijf in de map 'AppData\Local\Microsoft\Windows Sidebar\Gadgets' van het betreffende gebruikersprofiel. Iedere gadget heeft hier een eigen map, waarvan de naam op 'gadget' eindigt. Het van de grond af opbouwen van een gadget begint dan ook met het aanmaken van een dergelijke map. In ons voorbeeld wordt dat 'trein.gadget'. Deze naam geeft al aan wat voor soort gadget het gaat worden. Je geeft een treinstation op waar vandaan je wilt vertrekken en een treinstation waar je naartoe wilt reizen, en dan wordt regelmatig en automatisch bijgewerkt wanneer de eerstvolgende trein vertrekt. Deze reisinformatie komt van de NS zelf. Als extra service staat daar ook de aankomsttijd bij, en je kunt dat natuurlijk zelf uitbreiden met andere informatie als je dat wilt. Als je geen zin hebt om de code over te typen, kun je de gadget via de softlink downloaden.

Een verplicht onderdeel van iedere gadget is een manifest dat de inhoud nader beschrijft. Dit staat in het bestand manifest.xml, dat in de root van de gadgetmap moet staan. De opbouw daarvan staat in de listing linksonder. Voor eigen ontwikkelingen zijn met name de naam in regel 3 en welk HTML-bestand de gadget moet weergeven van belang. Dit laatste staat in regel 17 in het scr-attribuut van de base-tag. Het pictogram dat in regel 13 wordt genoemd, wordt door Windows gebruikt in de overzichtsweergave van de gadgets. Als dat ontbreekt, wordt een generiek pictogram gebruikt. De waarden van de tags version, author, copyright en description krijg je als gebruiker te zien als je in het gadgetoverzicht een gadget selecteert en dan op 'Details weergeven' klikt.

Het bestand met de eigenlijke gadgetcontent is op een paar kleinigheden na een normaal (X)HTML-bestand. De basis ervan staat in de tweede listing. De tag <g:background> in regel 9 komt wat ongewoon over. Dat is dan ook een gadgetspecialiteit voor het definiëren van een achtergrondafbeelding. Dat kun je ook zoals gebruikelijk met een background-attribuut van de body-tag of een van de omhullende divs in de stylesheet doen, maar Microsoft adviseert de hier getoonde structuur. Het is zelfs de enige mogelijkheid als er een achtergrondafbeelding met transparante delen gebruikt moet worden. Alleen daarmee kan een gadget er als een rechthoek met afgeronde hoeken uitzien of een volledig exotische vorm krijgen. Naast volledige transparantie is ook een alfakanaal mogelijk, bijvoorbeeld voor een slagschaduw, waar de desktop doorheen te zien is. In dat geval kun je het beste PNG als afbeeldingsformaat gebruiken.

Om ervoor te zorgen dat een gadget precies in de Sidebar van Windows Vista past, moet die exact 130 pixels breed zijn, en dat is inclusief een eventuele schaduw er omheen. Ook bij Windows 7 is die grootte nog gebruikelijk. Dan zijn de gadgets namelijk nog mooi onder elkaar te zetten. Om de opmaak wat te vergemakkelijken, moet je de totale grootte van de gadget in het stylesheetelement zetten dat verantwoordelijk is voor de body:

```
.gadgetbody
{
  width: 130px;
  height: 183px;
}
```

Voor de feitelijke content laat je dan een paar pixels rondom vrij om bijvoorbeeld niet per ongeluk op de schaduw te tekenen en daarmee de gewenste indruk te verstoren:

```
#gadgetContent
{
  position: absolute;
  top: 2px;
  left: 5px;
  width: 120px;
  height: 170px;
  overflow: hidden;
}
```

Met het manifest, het HTML-besatnd, de stylesheet en de achtergrondafbeelding zijn de minimumeisen voor een gadget klaar, en kun je het op de desktop laten zien. Dat doe

```
<?xml version="1.0" encoding="utf-8" ?>
<gadget>
  <name>c't - vertrektijden trein</name>
  <version>1.0</version>
  <author name="Noud van Kruysbergen">
    <info url="http://www.ct.nl" text="www.ct.nl" />
    <logo src="ctlogo48.png"  />
  </author>
  <copyright>2010 c't magazine</copyright>
  <description>Met deze gadget kun je zien wanneer de volgende trein
  gaat.</description>
  <icons>
    <icon height="64" width="64" src="ctlogo64.png" />
  </icons>
  <hosts>
    <host name="sidebar">
      <base type="HTML" apiVersion="1.0.0" src="gadget.html" />
      <permissions>Full</permissions>
      <platform minPlatformVersion="1.0" />
    </host>
  </hosts>
</gadget>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>Vertrektijden</title>
   <link rel="stylesheet" type="text/css" href="style.css">
   <script language="javascript" src="gadget.js"></script>
 </head>
 <body class="gadgetbody" onload="initGadget()">
   <g:background id="imgBackground" src="background.png">
     <div id="gadgetContent">

       <!-- hier staat de eigenlijke content -->


     </div>
   <g:background>
 </body>
</html>
```

**De basis van een HTML-pagina voor een gadget verschilt nauwelijks van wat gebruikelijk is voor een gewone webpagina.**

**Iedere gadget heeft een XML-bestand met een beschrijving, die onder andere aangeeft waar de bijbehorende HTML-pagina staat.**

je met een dubbelklik op het pictogram in het gadgetoverzicht. Om eventuele veranderingen te bekijken, is het voldoende om de gadget te sluiten en opnieuw te openen. Het gadgetoverzicht zelf hoef je alleen te sluiten als je het bestand gadget.xml hebt aangepast.

## Kwestie van instellen

Als de gadget de vertrektijden van een trein moet laden, moet je natuurlijk wel de mogelijkheid hebben om aan te geven vanaf welk station dat dan moet zijn. Bij gadgets is standaard al voorzien in een manier om via het steeksleutelpictogram rechtsboven naast de eigenlijke gadget rechtstreeks een dialoogvenster op te roepen. Als ontwikkelaar hoef je je alleen maar bezig te houden met de eigenlijke invoervelden, selectielijsten en dergelijke. Die definieer je in een XHTML-bestand. Die laten we hier uit ruimteoverwegingen niet helemaal zien, en eigenlijk staat er ook weinig spannends of raars in. De opbouw ziet er in principe zo uit:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    ...
    <link rel="stylesheet" type="text/css" href="style.css">
    <script language="javascript" src="settings.js">
    </script>
  </head>
  <body class="settingsbody">
    van station<br />
    <input type="text" name="station" id="station"
      size="30" maxlength="50">
    naar station<br />
    <input type="text" name="naar" id="naar"
      size="30" maxlength="50">
    ...
```

Om het vertrekstation in te typen wordt dus een gewoon input-element gebruikt, zoals je dat ook bij een website doet. Het is belangrijk dat ieder invoerelement een eenduidige id heeft, zodat je daar later met JavaScript makkelijk bij kunt.

Om ervoor te zorgen dat de gadget de dialoog kan laten zien, moet meegedeeld worden waar die te vinden is. Dat werkt met een paar regels JavaScript in de functie die automatisch aangeroepen wordt bij het onload-event in de HTML-pagina:

```
function initGadget()
{
  System.Gadget.settingsUI = "settings.html";
  System.Gadget.onSettingsClosed = settingsClosed;
}
```

De eerste regel in de functie definieert de instellingendialoog. De tweede zorgt ervoor dat de functie settingsClosed() automatisch opgeroepen wordt als de gebruiker de dialoog sluit. Hier kun je dan op veranderingen in de instellingen reageren.

De dialoog heeft zelf ook nog wat JavaScript nodig, in ieder geval minstens twee


**De gadgetomgeving heeft zelf een dialoogvenster voor eventuele opties.**

functies. De eerste slaat de door de gebruiker ingevoerde waarden op:

```
System.Gadget.onSettingsClosing = function(event)
{
  if (event.closeAction == event.Action.commit) {
    var sta = station.value;
    if(sta != "")
      System.Gadget.Settings.write("station", sta);
    var stn = naar.value;
    if(stn != "")
      System.Gadget.Settings.write("naar", stn);
    // ...
  }
}
```

Er wordt eerst gekeken of de gebruiker wel op OK heeft geklikt (event.Action.commit) en een vertrek- en aankomststation heeft ingevoerd (station.value en naar.value). Zo ja, dan slaat System.Gadget.Settings.write() de waarden in de instellingen van de gadget op onder de sleutels 'station' en 'naar'. Achter de coulissen schrijft Windows die in leesbare tekst in het bestand 'AppData\Microsoft\Windows Sidebar\Settings.ini' van het gebruikersprofiel – niet onbelangrijk om te weten als het om vertrouwelijke informatie gaat. De informatie verdwijnt daar weer uit als je de gadget op de desktop sluit. Voor ontwikkelaars betekent dit dat de invoer na het herstarten van een gadget weer herhaald moet worden of dat je in de testfase de functie initGadget() met een paar regels code uitbreidt om standaardwaarden in te stellen.

Een tweede functie in de code voor de dialoog vult de invoervelden met de waarden die de gebruiker de laatste keer heeft ingevoerd.

## Beweging

Voor het oproepen van de vertrektijden van de treinen maken we gebruik van een


**De hier beschreven demogadget laat zien wanneer de volgende trein van een bepaald station naar een ander station vertrekt.**

truc. We gebruiken Internet Explorer als een ActiveX-object, besturen die op afstand en dan halen we de gewenste informatie uit de webpagina. Deze webpagina is van de mobiele website van de Nederlandse Spoorwegen. Een mobiele website is vaak eenvoudiger gestructureerd en laadt grafisch vaak minder overhead in dan een reguliere website, die voor een pc bedoeld is. Als je deze gadget als uitgangspunt voor je eigen ontwikkelingen neemt, moet je er in ieder geval op letten of de aanbieder van de informatie die je wilt gebruiken een dergelijke site beschikbaar heeft.

De gadget doet in principe op afstand het volgende met Internet Explorer. Hij opent de pagina http://m.ns.nl, vult de invoervelden in met de gewenste gegevens, verstuurt de pagina en kijkt op de resultaatpagina naar de antwoorden. Daar kun je makkelijk nog allerlei stappen aan toe- en tussenvoegen om bijvoorbeeld aan te melden bij een inlogpagina en aan het eind weer uit te loggen.

Het aanroepen van de invoerpagina ziet er in JavaScript als volgt uit:

```
ie = new ActiveXObject("InternetExplorer.Application");
ie.visible = true;
ie.Navigate("http://m.ns.nl/planner.action");
```

De regel ie.visible = true is alleen voor de testfase bedoeld, daarmee kun je in de gaten houden wat de op afstand bestuurde Internet Explorer aan het doen is. Als alles naar wens werkt, kun je deze regel verwijderen of op false zetten.

Om vervolgens uit te vissen wat er daarna moet gebeuren, is het aan te raden de website waar de informatie vandaan moet komen eerst handmatig met Internet Explorer op te roepen. Doe dat zeker ook als je normaal met een andere browser werkt, omdat sommige websites op hun pagina's verschil maken tussen de diverse browsers en bovendien kunnen de bovengenoemde Ontwikkelhulpprogramma's van IE8 een waardevolle hulp zijn bij het analyseren van een website – al kan dat bijvoorbeeld ook met de extensie Firebug voor Firefox. Om bijvoorbeeld te achterhalen hoe het invoerveld voor het vertrekstation heet, kies je bij de Ontwikkelhulpprogramma's het menuitem 'Zoeken / Element selecteren door te klikken'. Vervolgens klik je op het invoerveld waar je het vertrekstation invult, waarna in de broncode de HTML-tag voor het invoerveld tevoorschijn komt – in dit geval heeft die gelukkig een eenduidige id, waarmee de invoer snel is gedaan:

```
var doc = ie.document;
var input = doc.getElementById("from");
if(!input) {
  setErrorText("invoer niet mogelijk");
```

```
  ie.Quit();
  ie = null;
  return;
}
input.value = System.Gadget.Settings.read("station");
```

Met getElementById() vind je het invoerveld – zo niet, dan is er bij het oproepen van de website iets verkeerd gegaan. De functie setErrorText() is een eigen brouwsel. Het uitlezen van de stationsnaam uit de instellingen heb je boven al eerder gezien.

Ook het invoerveld van het aankomststation heeft een eigen id, die niet geheel onverwacht to is. Bij de datum en de tijd hoeven we zelf niet veel te doen, die worden door de mobiele website van de NS vanzelf op de huidige datum en tijd gezet. En ook hoeven we niet te kiezen tussen vertrek- of aankomsttijd, omdat die automatisch op 'Vertrekken' staat. Als je de gadget uit wilt breiden naar aankomsttijden, hoef je alleen de status van deze radiobuttons om te draaien.

Om het formulier te versturen, moeten we het klikken op de knop 'Reisadvies' nog simuleren. Uit de broncode van de website blijkt dat dit de submit-button <input name="planroute" type="submit" value="Reisadvies"/> is. Deze knop heeft geen id, en is met getElementById dan ook niet te benaderen. De knop heeft wel een name, maar er bestaat geen instructie met de naam getElementByName. Er is wel het attribuut getElementsByName, waarbij het verschil in de s zit. Vandaar de volgende constructie:

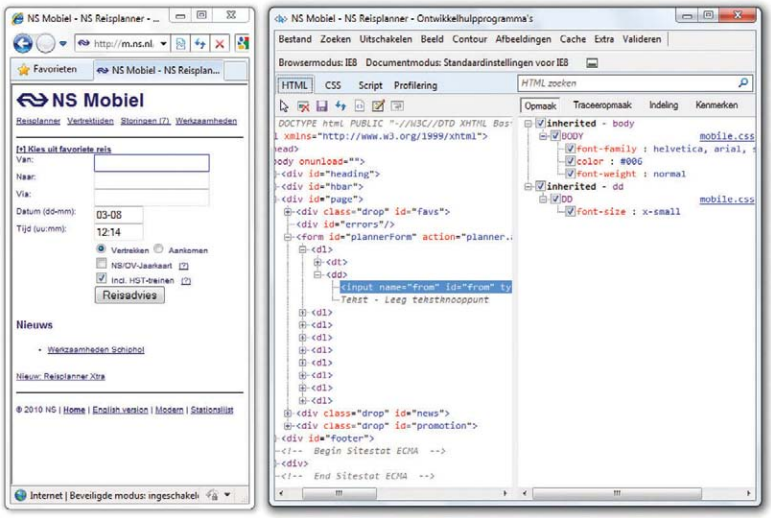
```
var inputs = doc.getElementsByName("planroute");
for (var i = 0; i < inputs.length; i++) {
  if (inputs[i].value == "Reisadvies") {
    inputs[i].click();
    break;
  }
}
```

Daarbij krijgen we alle elementen met de naam planroute, maar dat is er in principe maar één. Voor de volledigheid doorlopen we ze allemaal om te kijken of de waarde ervan Reisadvies is, maar je zou ook meteen inputs[0].click() kunnen doen.

Daarmee is het formulier verzonden en komt de antwoordpagina binnen. Vanaf dat punt wordt het een beetje vies. Het blijkt dat de informatie waar het om gaat voornamelijk in de cellen van een tabel staat, vandaar dat we op zoek gaan naar alle elementen met de HTML-tag td:

```
var td = doc.getElementsByTagName("td");
for (var d = 0; d < td.length; d++) {
  text += td[d].innerText + "<br />";
}
```

De teksten die in een cel staan, worden via het attribuut innerText één voor één ingelezen en op een eigen regel gezet. Daar kun je verder natuurlijk nog van alles mee doen, maar in dit geval worden ze alleen getoond zoals ze op de website staan. Er zijn voor andere websites nog allerlei andere manieren om daar informatie vanaf te halen, en dan is het vaak een kwestie van fantasie om die daar ook uit te halen.




**Bij het analyseren van een website kunnen de Ontwikkelhulpprogramma's van IE8 een waardevolle hulp zijn.**

## Haastige spoed

In de broncode van de gadget staan alle hier getoonde codes niet allemaal in één functie, maar zijn die verdeeld over meerdere functies. Dat heeft naast de overzichtelijkheid nog een extra reden: je kunt het uitlezen van document uit het IE-object niet meteen beginnen na het aanroepen van Navigate() of click(), maar je moet daarmee wachten totdat Internet Explorer de pagina helemaal geladen heeft. Nu is JavaScript er helemaal niet voor gemaakt om te wachten – een Sleep() of Wait() functie ontbreekt. Dan blijft er niets anders over dan via window.setInterval() een timer in te stellen die om de zoveel milliseconden een functie oproept die via de eigenschap ie.busy controleert of Internet Explorer nog met laden bezig is. Pas als dat niet meer het geval is, kan de timer weer ophouden en kan de volgende fase van de verwerking beginnen.

Om ervoor te zorgen dat dit enigszins makkelijk en flexibel te doen is, staat er in de gadgetbroncode een IEStager-object. Deze bevat een lijst van acties en wacht in de tussentijd tot Internet Explorer niets meer te doen heeft. De update()-functie maakt een dergelijk object nadat IE met Navigate() naar de invoerpagina is gestuurd. Er wordt een lijst met stappen meegegeven die vervolgens uitgevoerd moeten worden:

```
var actions = new Array("enterQuery(this)",
                        "readAnswer(this)");
var stager = new IEStager(actions);
stager.proceed();
```

Met proceed() begint de stager te wachten en vervolgens de volgende functie uit de lijst op te roepen. Om ervoor te zorgen dat dit werkt, moet hij van actie naar actie worden gestuurd. En dat is de reden dat de functies enterQuery() en readAnswer() ieder een stager als parameter verwachten en ze in de listing met een this als argument staan.

## Uitvoer

Wat dan nog ontbreekt, is het bewerken van de informatie om in de gadget getoond te worden. Dat heeft wel een aantal beperkingen, omdat de venstergrootte van een gadget strak begrensd is. Als je dan net als in dit voorbeeld niet van tevoren kunt voorspellen hoeveel informatie er binnenkomt in verband met spoorwegwerkzaamheden of tussenstations, zal een deel van de informatie niet te zien zijn. Je kunt het je dan makkelijker maken door niet alle HTML-code met JavaScript te maken, maar van tevoren velden in een HTML-bestand aan te maken en die door het script dan met innerText te laten vullen. Vaak wordt de informatie een stuk duidelijker door het aloude adagium 'een beeld zegt meer dan duizend woorden'.

De laatste stap is dan om van alle code, afbeeldingen en andere onderdelen een complete gadget te maken, die je dan aan anderen kunt geven en die op een andere computer met een dubbelklik te installeren is. Dat is niet al te moeilijk: je stopt alle bestanden in een ZIP- of CAB-bestand en verandert de extensie in .gadget.

Het gadgetplatform heeft nog een aantal andere functies, die hier niet aan bod zijn gekomen, voor bijvoorbeeld het uitlezen van de systeemtoestand van de pc of het besturingssysteem. Je kunt ook het beeldscherm uitbreiden met zogeheten flyouts. Als je dieper in deze materie wilt duiken, staan er in de MSDN Library [1] inleidende tutorials en gedetailleerde referentie-informatie bij het onderwerp Windows Sidebar. In de genoemde gadgetgalerie op de webserver van Microsoft staan een paar echte mooie gadgets, waar je je door kunt laten inspireren. Na het installeren staat de broncode immers op de harde schijf. (nkr)

**Literatuur**

[1] Windows Sidebar in de MSDN Library: http://msdn.microsoft.com/library/aa965850.aspx

Softlink 1010108 c't