Att anropa webbservern inifrån en webbsida

I den här artikeln vill jag demonstrera ett sätt att hämta information ifrån en webbserver utan att behöva ladda om hela sidan. Detta kan användas till mycket, alltifrån att verifiera inmatningar till att kontrollera om några nyheter har inkommit. (Jag kommer i en framtida artikel visa på hur man kan utveckla det senare med ett snyggt popup-fönster som dyker upp när en nyhet har publicerats på servern.)

Av
Dag König
dag.konig@knowit.se
Konsult inom Microsoft .NET på Know IT

Exemplet, som går att ladda ned , är ett ASP.NET-projekt som består av fyra AspX-filer. `Start.AspX` är huvudfilen och det är den som innehåller alla exempel. Jag kommer att visa tre olika exempel på denna teknik. Dessa tre är:

- Hämta ett värde ifrån servern och visa det i webbläsaren.
- Validera ett inmatningsfält med kod som körs på servern.
- Fråga servern regelbundet efter information. ("polla servern")

1. Hämta ett värde ifrån servern och visa det i webbläsaren

Klientsidan

I filen `Start.AspX` skapas tre olika html-taggar. En knapp, ett div-element för att visa resultatet och ett xml-element (XML Data Island) för att kunna hämta och tolka en xml-sträng.

```
<xml id="xmlAnswer1" ondataavailable="responseAnswer1()"/>

<input type="button" onclick="requestAnswer1()"
    value="Test One">
    <div id="divAnswer1"></div>
```

Funktionen `requestAnswer1()` startas med hjälp av knappens `onClick`-händelse.

```
function requestAnswer1() {
    xmlAnswer1.src =
    "http://localhost/UpdateInBackground/GetAnswer1.AspX";
}
```

Den funktionen använder sig av xml-objektets egenskap `src` för att initiera en förfrågan ifrån webbservern. (Observera att jag har hårdkodad url:en. Den kan behöva ändras för att få exemplet att fungera hos dig.)

I xml-elementet har jag talat om att funktionen `responseAnswer1()` skall köras när ny information blir tillgänglig i objektet. När den blir tillgänglig så utförs händelsen `onDataAvailable`.

```
function responseAnswer1() {
    var oNode = xmlAnswer1.XMLDocument.selectSingleNode("answer");
    divAnswer1.innerText = oNode.text;
}
```

Den funktionen läser ut svaret ur xml-strängen och skriver ut svaret I webbläsaren.

Serversidan

Den uppmärksamma såg att jag i `requestAnswer1()` anropade sidan `GetAnswer1.AspX`. Innehåller i den filen är mycket enkelt. Code-behind-filen ser ut på följande sätt:

```
Private Sub Page_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load

    Response.ContentType = "text/xml"
    Response.Write("<answer>Donald</answer>")
End Sub
```

Det enda som den gör är att ändra `ContentType` till xml och sedan skriva ut en giltig xml-sträng. Jag rekommenderar att man använder sig av xml-klasserna som finns i .NET om mer avancerade xml-strängar skall genereras.

Själva AspX-filens innehåll är ändå enklare. Den består endast av siddirektivet, `@Page`.

```
<%@ Page Language="vb" AutoEventWireup="false"
    Codebehind="GetAnswer1.aspx.vb"
    Inherits="UpdateInBackground.GetAnswer1"%>
```

Denna rad krävs för att sidan skall hitta sin code-behind-fil, eller rättare sagt, att den skall vet ifrån vilken klass som den skall ärva. I det här fallet klassen `UpdateInBackground.GetAnswer1`.

Observera att det är mycket viktigt att man tar bort all annan kod ifrån den här sidan. Sidan kommer annars att generera htmlkod istället för ren xml.

2. Validera ett inmatningsfält med kod som körs på servern

Klientsidan

Vi har ett inmatningsfält och en knapp. Ett anrop sker till servern när knappen aktiveras. Med anropet medföljer innehållet i inmatningsfältet.

Servern kontrollerar värdet och skickar tillbaka ett svar. En `submit()` sker om värdet är giltigt. En text visas för användaren om det å andra sidan är ogiltigt. Koden i html-sidan ser ut på följande sätt:

```
function requestAnswer2() {

    xmlAnswer2.src =
        "http://localhost/UpdateInBackground/GetAnswer2.AspX?Name=" +
        document.all.TextBoxName.value;
}

function responseAnswer2() {
    var oNode = xmlAnswer2.XMLDocument.selectSingleNode("answer");
    if(oNode.text == 'ok')
        document.forms[0].submit();
    else
        document.all.TextBoxName.value = "Name Invalid";
}
```

```
<xml id="xmlAnswer2" ondataavailable="responseAnswer2()"/>

Input Name: <input type="text" id="TextBoxName"/>
<input type="button" value="Check Name" onclick="requestAnswer2()" >
```

Det ser alltså inte så annorlunda ut emot det tidigare exemplet. Den nya är att värdet skickas med till servern via querysträngen.

Vi går händelserna i förväg och ser redan nu på svaret på förfrågan som gjordes i `requestAnswer2()`. Som i förra exemplet läses först svaret ut ifrån xml-strängen. Programmet kommer att posta hela sidan (via metoden `submit()`) om valideringen var ok. Om valideringen istället signalerar att den inmatningen var felaktig, kommer texten ”Name Invalid” att skrivas ut i inamtningsfältet.

Serversidan

Sidan `GetAnswer2.AspX` validerar det inmatade.

```
Private Sub Page_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load

    Response.ContentType = "text/xml"
    Dim l_name As String = Request.QueryString("Name")
    Dim l_answer As String

    Select Case l_name.ToLower
        Case "don", "monica", "robin"
            l_answer = "ok"
        Case Else
            l_answer = "invalid"
    End Select

    Response.Write("<answer>" & l_answer.ToString.ToLower & "</answer>")
End Sub
```

Vi ser av koden att det här inmatingsfältets enda godkända värden är Don, Monica eller Robin. Ok kommer att skickas tillbaka om någon av dessa strängar har angetts, annars kommer istället Invalid att skickas tillbaka.

Kom ihåg att Aspx-filen endast skall innehålla @Page-direktivet.

3. Fråga servern regelbundet efter information

Det sista exemplet, som jag tycker är mest intressant, ger oss möjlighet att fråga servern, i bakgrunden, om det har kommit någon ny information som användaren behöver få kännedom om. Detta öppnar möjligheter till en massa roliga möjligheter.

Det här exemplet är kraftigt förenklat. Jag kommer endast att generera ett tal på serversidan och skicka tillbaka det vid varje förfrågan.

Klientsidan

```
var l_timer;

function requestAnswer3() {
    xmlAnswer3.src="http://localhost/UpdateInBackground/GetAnswer3.AspX";
}

function responseAnswer3() {
   var oNode = xmlAnswer3.XMLDocument.selectSingleNode("answer");
   divRandomNumber.innerText = oNode.text;
}
```

```
<xml id="xmlAnswer3" ondataavailable="responseAnswer3()"/>

<input type="button" value="Start"
    onClick="l_timer=setInterval('requestAnswer3()', 3000);"/>
<div id="divRandomNumber"></div>
<input type="button" value="Stop" onClick="clearInterval(l_timer);"/>
```

Skriptdelen av sedan innehåller inga nya grejor.

Html-delen innehåller, förutom det nu välkända elementet xml, tre kontroller. Två knappar, en för att starta förfrågningarna och en för att stoppa dessa. Div-elementet använts för att skriva ut innehållet ifrån det som kommer ifrån servern.

I startknappens händelse `onClick` används metoden `setInterval()` för att skapa en Timer. Den tar två parametrar. Den första talar om vilken metod som skall anropas när timern aktiveras och den andra talar om timerns tidsintervall i millisekunder. Metoden returnerar ett värde som lagras i variabeln `l_timer`. Detta är en referens till denna Timer.

Stoppknappens händelse `onClick` stoppar Timern genom att anropa `clearInterval()` med parametern `l_timer`.

Serversidan

Det enda som görs är att slumptal skapas och skickas tillbaka.

```
Private Sub Page_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load

    Response.ContentType = "text/xml"

    Response.Write("<answer>" & New Random().Next(1, 9999).ToString & _
        "</answer>")
```

```
End Sub
```

Kom ihåg att Aspx-filen endast skall innehålla @Page-direktivet.

Sammanfattning

Alla dessa tre exempel är implementerade i samma webbsida. Dessa går att köra samtidigt. Man kan alltså samtidig hålla på att validera inmatningsfält medan en kontroll om det har kommit något nytt ifrån servern.

Jag skall i nästa artikel fördjupa det tredje exemplet och visa hur man kan bygga en väldigt snygg popup som visas på skärmen när ny information har publicerats på servern.

Disclaimer

Det här kodexemplet är mycket förenklat och skrivet utifrån detta perspektivet. Detta kanske inte passar som produktionskod. Jag tar inte på mig något ansvar för implementeringen av den. Tänk själv och testa. Jag tar inte heller på mig någon form av support av koden, men kommentarer på förbättringar emottages tacksamt.