

# Set Focus to a Control in ASP.NET with Delphi 2005

By Bob Swart

For the past few weeks, I've been working on a little ASP.NET project in Delphi 2005 (migrating from IntraWeb 5.x), and I've had to "invent" some techniques that in retrospect also might be useful to share with other ASP.NET developers - using Delphi 2005, Delphi 8 for .NET or any other .NET development language or environment for that matter.

## SetFocus

First of all, there were a few pages where a DropDownList was used (with a set of values). The application needed to respond directly if a new selected item in the DropDownList was picked. That was easy enough by setting the AutoPostBack property of the DropDownList to True, writing some event handling code in the DropDownList\_SelectedIndexChanged method. Although this indeed performed a round-trip to the (web) server, the downside was that the focus on the DropDownList was lost. This may not sound like a big deal, but can quickly become a major annoyance if you just want to scroll down in the DropDownList, see the changes of the new choice, and after every change have to click on the DropDownList again to give it the focus.

Unfortunately, there is no SetFocus method or ActiveControl property in ASP.NET, so I had to think of another solution. The easiest way is to generate some JavaScript and add it to the ASP.NET Page. There is a special method called RegisterStartupScript which can be used to add JavaScript (or any other scripting code) to the startup section of the Page, which is the ideal place for setting the focus. So, I knew how to add the JavaScript to the Page, now all I had to do was write the right JavaScript.

There is a focus() method in JavaScript, but in order to call it, I need to know the ID of the control for which I'm calling it. Let's assume the DropDownList has its ID set to ddlNames, then the call is ddlNames.Focus(). That's not enough just yet, since we must also specify the ID of the Form on which ddlNames is placed. If you take a look at the Form in an ASP.NET Page that's generated by Delphi 2005, you'll notice that it has no associated ID, so you must assign one yourself. Click on the Form in the HTML Designer, and then specify a value for the id property in the Object Inspector. Sometimes the Delphi 2005 Object Inspector doesn't show any properties for the Form, in which case you can use the tag editor, and manually change the Form tag to the following:

```
<form id="frmWhatever" runat="server">
```

When you're done, make sure that there is a declaration of the following form in your ASP.NET Page class definition:

```
frmWhatever: System.Web.UI.HtmlControls.HtmlForm;
```

For some reason, the Delphi 2005 HTML Designer doesn't always produce this declaration for me. Obviously a more useful name than frmWhatever can be used here, but you get the idea.

Once we have given both the Form and the DropDownList an ID, we can produce the JavaScript which should be as follows:

```
<script code="JavaScript">
<!--
    document.frmWhatever.ddlNames.focus();
-->
</script>
```

This will work in Internet Explorer, but apparently not in all other browsers. When I look at the generated source for the ASP.NET Page (in the browser, after the Page is generated, do View | Source), I see a construct with the following contents:

```

<script language="javascript" type="text/javascript">
<!--
function __doPostBack(eventTarget, eventArgument) {
    var theform;
    if (window.navigator.appName.toLowerCase().indexOf("microsoft") > -1) {
        theform = document.frmWhatever;
    }
    else {
        theform = document.forms["frmWhatever"];
    }
    ...
}

```

This leads me to believe I should also add the check for the Browser to my JavaScript code, leading to the following snippet (since I only call one method, there's no need for the "theform" variable here):

```

<script code="JavaScript">
<!--
if (window.navigator.appName.toLowerCase().indexOf("microsoft") > -1) {
    document.frmWhatever.ddlNames.focus();
} else {
    document.forms["frmWhatever"].ddlNames.focus();
};
-->
</script>

```

Producing this JavaScript in Delphi 2005 and passing it to the RegisterStartupScript is now easy. You can either hardcode the ID's of the Form and DropDownList (as I've done above), or use the variables themselves and use their ClientID property, as I'm doing in the generic code below:

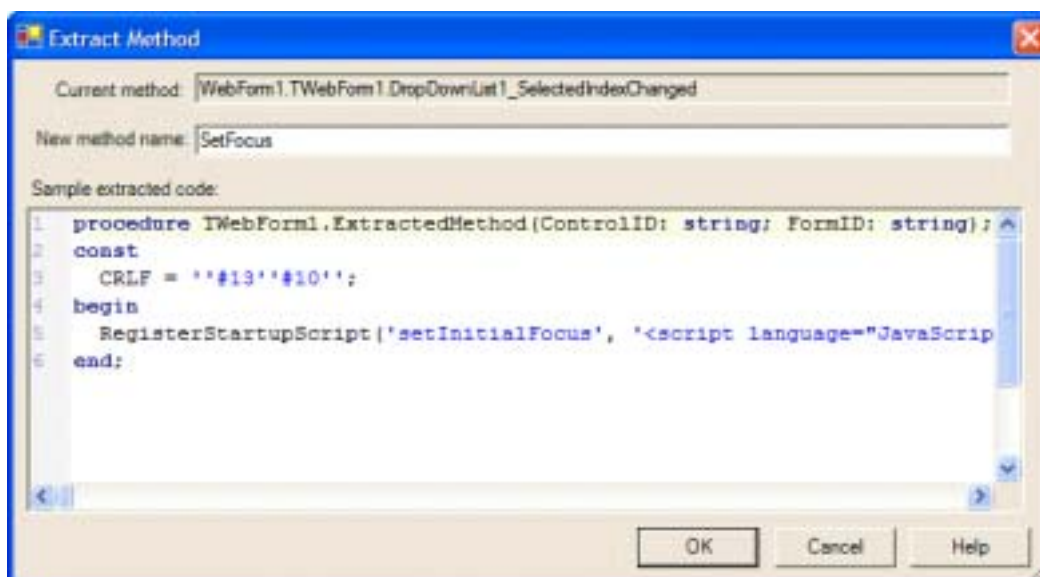
```

RegisterStartupScript('setInitialFocus',
  '<script language="JavaScript">' + CRLF +
  '<!--' + CRLF +
  '  if (window.navigator.appName.toLowerCase().indexOf("microsoft") > -1) { ' + CRLF +
  '    document.' + frmWhatever.ClientID + '.' +
  '    ddlNames.ClientID + '.focus(); ' + CRLF +
  '  } else { ' + CRLF +
  '    document.forms["' + frmWhatever.ClientID + '"].' +
  '    ddlNames.ClientID + '.focus(); ' + CRLF +
  '  }; ' + CRLF +
  '//-->' + CRLF +
  '</script>' + CRLF)

```

## Refactor!

Assuming you want to use this code in more than one place in your application, it's even more generic to put the code in a special routine. In the Delphi 2005 Code Editor, select the code we've just written (shown above), right-click with the mouse and from the Refactoring menu select "Extract Method". This will produce the following dialog:



You may want to change the constant definition for CRLF to a simple #13#10 afterwards (and you may also want to change the order of the arguments, but that's only a matter of taste).

With a little manual reformatting, the final routine can be seen below:

```
procedure TWebForm1.SetFocus(ControlID: string; FormID: string);
const
  CRLF = #13#10;
begin
  RegisterStartupScript('setInitialFocus', '<script language="JavaScript">' + CRLF +
    '<!--' + CRLF +
    '  if (window.navigator.appName.toLowerCase().indexOf("microsoft") > -1) { ' + CRLF +
    '    document.' + FormID + '.' + ControlID + '.focus(); ' + CRLF +
    '  } else { ' + CRLF +
    '    document.forms["' + FormID + '"].' + ControlID + '.focus(); ' + CRLF +
    '  }; ' + CRLF +
    '//-->' + CRLF +
    '</script>' + CRLF);
end;
```

The method SetFocus can now be called for any control on your form. An example where I called it is as follows:

```
SetFocus(frmWhatever.ClientID, ddlNames.ClientID);
```

For an even more useful method, you can also pass the Page itself, and move the entire routine outside the TWebForm1 class - for example in its own WebUtils unit (but that's left as exercise for the reader).

## Summary

Using the AutoPostBack property on ASP.NET controls like a DropDownList ensures that you can perform a round-trip to the server whenever the selected item is changed. Normally, however, the DropDownList would not regain the focus when the page is displayed again, something which can be quite annoying (especially if you want to browse quickly through the alternatives listed in the DropDownList).

In this article, I've shown you the JavaScript required, as well as a generic Delphi routine to produce the customised JavaScript inside your ASP.NET web pages. This will make sure the DropDownList retains the focus, and has also increased my ASP.NET user interface experience (and that of my clients).



*Bob Swart (aka Dr Bob - [www.drbob42.net](http://www.drbob42.net)) is a software developer, author, trainer, consultant and webmaster for his own one-man company, Bob Swart Training & Consultancy in Helmond, The Netherlands. He writes for numerous computing magazines as well as his own training material, and is also webmaster to the group.*