Fighting Bots

Christian Wenz

Overview

Automated bots plaster weblogs and other websites with spam, submitting comment forms without any user interaction. The NoBot control in the ASP.NET AJAX Control Toolkit can help fight those bots.

Steps

One common approach to defeat bots is to use CAPTCHAs—Completely Automated Public Turing test to tell Computers and Humans Apart. A Turing test was originally a test where someone needed to decide whether a communication partner is a human or a machine. In the web, a CAPTCHA usually consists of an image with some distorted letters on it. The idea is that only a human can read the letters on the image, whereas OCR algorithms will fail.
There are several advantages and disadvantages to this approach, but a discussion of this is beyond the scope of this tutorial. There is however a control in the ASP.NET AJAX Control Toolkit which provides a similar approach: **NoBot**. It is easier to overcome than a CAPTCHA, but is very easy to use and fares extremely well on websites like blogs where it is considered a success if most spam attempts are defeated, which the **NoBot** control can do.

NoBot intercepts the postback of the current ASP.NET web form if at least one of these conditions is met:

- The browser fails to solve a JavaScript puzzle (for instance when JavaScript is deactivated)
- The user submitted the form to fast
- The client IP address submitted the form too often in a certain period of time.

In order to check for these conditions, the **NoBot** control requires these attributes (all of them optional):

- **ResponseMinimumDelaySeconds**—minimum amount of seconds between postbacks
- **CutoffWindowSeconds**—length of time interval in which postbacks from one IP are measures
- **CutoffMaximumInstances**—maximum amount of seconds per time interval

The following markup demands that at least two seconds elapse between postbacks and that there are only five postbacks or less within a 30 seconds interval:

```
<ajaxToolkit:NoBot ID="nb" runat="server"
   CutoffMaximumInstances="5" CutoffWindowSeconds="30"
  ResponseMinimumDelaySeconds="2" />
```

Then—as usual—make sure to include the **ScriptManager** in the page so that the ASP.NET AJAX library is loaded and the Control Toolkit can be used:

```
<asp:ScriptManager ID="asm" runat="server" />
```

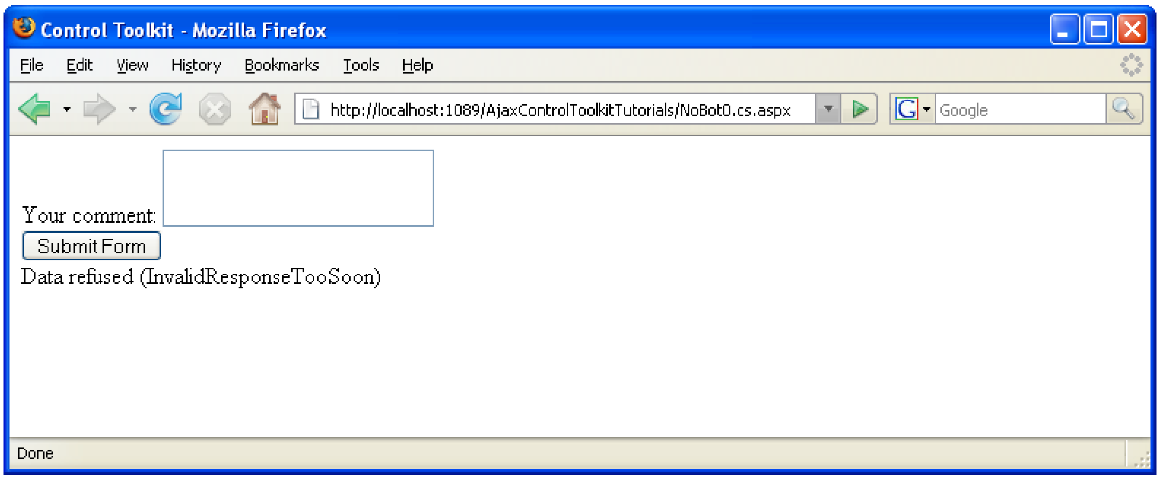
Since most of the checks **NoBot** is doing occur on the server side, you need to check the result of these validations. This can be done by calling **NoBot**'s **IsValid()** method. It has one argument (as an **out** parameter/**ByRef** parameter) which is of type **NoBotState**. Its string representation contains the reason when the check fails and **Valid** otherwise. The following code outputs a message according to **NoBot**'s result:

```
<script runat="server">
  void Page_Load()
  {
    if (Page.IsPostBack)
    {
      NoBotState state;
      if (!nb.IsValid(out state))
      {
        Label1.Text = "Data refused (" +
   HttpUtility.HtmlEncode(state.ToString()) + ")";
      }
      else
      {
        Label1.Text = "Data entered.";
      }
    }
  }
</script>
```

Finally, you need a form to submit—and a label element to output the message, and you are done!

```
Your comment:
<asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine"
   /><br />
<input type="submit" id="Submit1" runat="server" value="Submit
   Form" /><br />
<asp:Label ID="Label1" runat="server" />
```

When you run this script and deactivate JavaScript or submit the form within the first two seconds or submit the form seven times within thirty seconds, you will get an error message. However use this control wisely, since only about 90-95% of users have JavaScript activated, therefore 5-10% of users will fail **NoBot**'s test.


Control Toolkit - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:1089/AjaxControlToolkitTutorials/NoBot0.cs.aspx
Google
Submit Form
Done

Your comment:

Data refused (InvalidResponseTooSoon)

This error message could have been caused by a bot