Customizing the BlaiseIS XSLT Stylesheets and ASP Scripts

Edwin de Vet (CentERdata, Tilburg, The Netherlands)

Arnaud Wijnant (CentERdata, Tilburg, The Netherlands)

1 Abstract

At CentERdata we routinely use BlaiseIS to administer questionnaires over the web to a large number of respondents. We often have to customize the BlaiseIS system to meet the requirements of our clients. These customizations almost always involve modifiying the XSLT sytlesheets that are used for rendering the HTML pages. This paper contains examples of how we extended the capabilities of BlaiseIS. These examples include incorporating a slider using jQuery, using the HTML tag “label” with checkboxes and radio buttons to make the text clickable, styling of tables etc. Besides modifying the existing BlaiseIS stylesheet, we created also a lightweight XSLT stylesheet from scratch. This stylesheet creates simple, lean HTML while still providing the functionality we need. We found that this stylesheet solves some performance issues we encountered with large tables. Furthermore, it served as a basis for our stylesheet for mobile devices. We also modified ASP scripts to add functionality. Examples are the key stroke logs in text format introduced in the page handler and a simple security check using the SHA1 algorithm in the interview starter.

2 Introduction

At our institute we run web questionnaires using BlaiseIS for a large number of clients. Since our migration to BlaiseIS in 2010, we gained a lot of experience in adapting the system to our needs and the needs of our customers. These modifications to BlaiseIS almost always involve adapting the XSLT style sheets and/or the asp scripts. This paper will give a broad overview of those modifications. We run Blaise 4.8.2 on our servers.

3 XSLT Style Sheets

3.1 Clickable text

We use the HTML tag “label” for text belonging to checkboxes and radio buttons. This has the advantage that respondents can select or deselect those elements by clicking on the text. Previously, we added an Id to the HTML input elements of type radio and checkbox. This Id is the same as the Id of the corresponding CategoryControl element in the XML used as source for the transformation. The template RichTextElement (in biSimpleHTMLWebPage.xsl) was modified so that it adds a label around the text with a “for” attribute that references the corresponding checkbox/radio button. This is a simplified example of the HTML:

```
<input name="qu1fpala" id="qu1fpala1" value="1" type="radio">
<label for="qu1fpala1">Option A</label>
<input name="qu1fpala" id="qu1fpala2" value="2" type="radio">
<label for="qu1fpala2">Option B</label>
```

3.2 Slider

To create sliders in BlaiseIS questionnaires, we used the jQuery-UI library. The following code was put in the Blaise source file:

```
<div id=""slider2""></div>
<script type=""text/javascript"">
  $(function() {
       createSlider(2);
  });
</script>
```

The XSLT style sheet was modified so that it included the jquery-1.3.2.min.js and jquery-ui-1.7.2.custom.min.js libraries, a CSS file for the slider and the JavaScript code for the createSlider function:

```
function createSlider(id) {
       var element = document.getElementById("qu" + id + "_id");
       element.style.display = 'none';
       var val = 550;
       if (element.value != '') {
              val = parseInt(element.value);
       } else {
              element.value = val;
       }
       var sliderOpts = {
                min: 100,
                max: 1000,
                value: val,
                slide: function(e, ui) {
                     element.value = ui.value;
                }
              };
       $("#slider" + id).slider(sliderOpts);
}
```

This function creates a slider with values between 100 and 1000 and a default value of 550 (unless the answer is already present in the form). In the Blaise source file, the data type is integer. The createSlider function makes the open text field invisible to the user. By moving the slider, this hidden element is filled with the set value of the slider and once the form is submitted, the selected value is stored in the Blaise database.

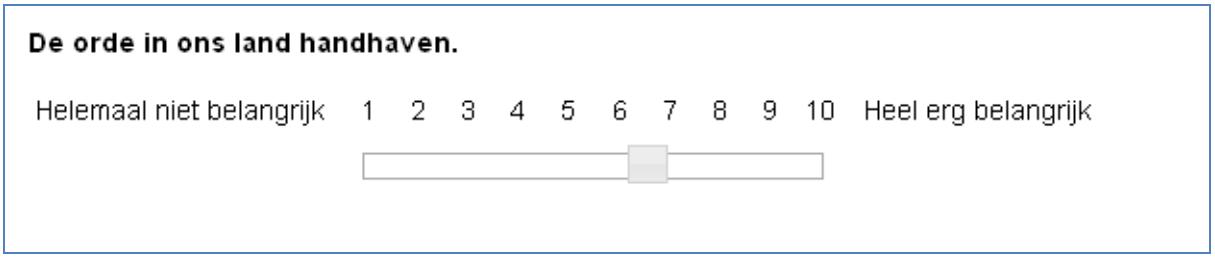

Figure 1: jQuery slider in a BlaiseIS questionnaire.

3.3 Table Styling

In our questionnaire we use a lot of tables/grouped questions. The XSLT style sheet was modified in such a way that the header row (<tr> HTML element) always gets the CSS class “HeaderRow”. The other

rows in the table (containing the questions) get alternating class “OddRow” and “EvenRow”. Standard CSS can be used to give each type of row its special style.

	1	2	3	4	5
Vond u het moeilijk om de vragen te beantwoorden?	○	○	○	○	○
Vond u de vragen duidelijk?	○	○	○	○	○
Heeft de vragenlijst u aan het denken gezet?	○	○	○	○	○
Vond u het onderwerp interessant?	○	○	○	○	○
Vond u het plezierig om de vragen in te vullen?	○	○	○	○	○

Figure 2: Example of table styling using the “HeaderRow”, “OddRow” and “EvenRow” CSS classes.

3.4 Simple Style Sheet / Mobile Style Sheet

At CentERdata we also created a minimal XSLT style sheet that only has the functionality we require. This style sheet is roughly 6 times smaller than the original style sheets, renders much faster and is easier to modify due to its simplicity [1]. This style sheet formed the basis for a style sheet for mobile devices and is documented extensively elsewhere [2].

3.5 Custom questions

Adding interactive questions can enhance the respondents’ experience. For this, we created a mechanism to easily integrate web applications in a questionnaire programmed in Blaise[3]. This opens up the opportunity to make use of all the options the web has to offer for responding to a questionnaire item. Examples of this are special interfaces (like auto complete boxes and even games), interactive feedback to respondents, and the use of other resources available on the web.

3.6 Trigram search/encode question

One of the custom questions that we created is an automatic look-up and encode question. This question uses a database that contains descriptions in which a respondent can search for items and attached to that a code that will be used for the answer in the Blaise questionnaire.

Respondents that visit this question can simply start typing their answer and then select the correct answer from a list of possibilities that is very similar to a Google search suggestion box. Once the respondent has selected the answer, the application encodes the answer automatically on the background and the routing of the questionnaire can be based on this code.

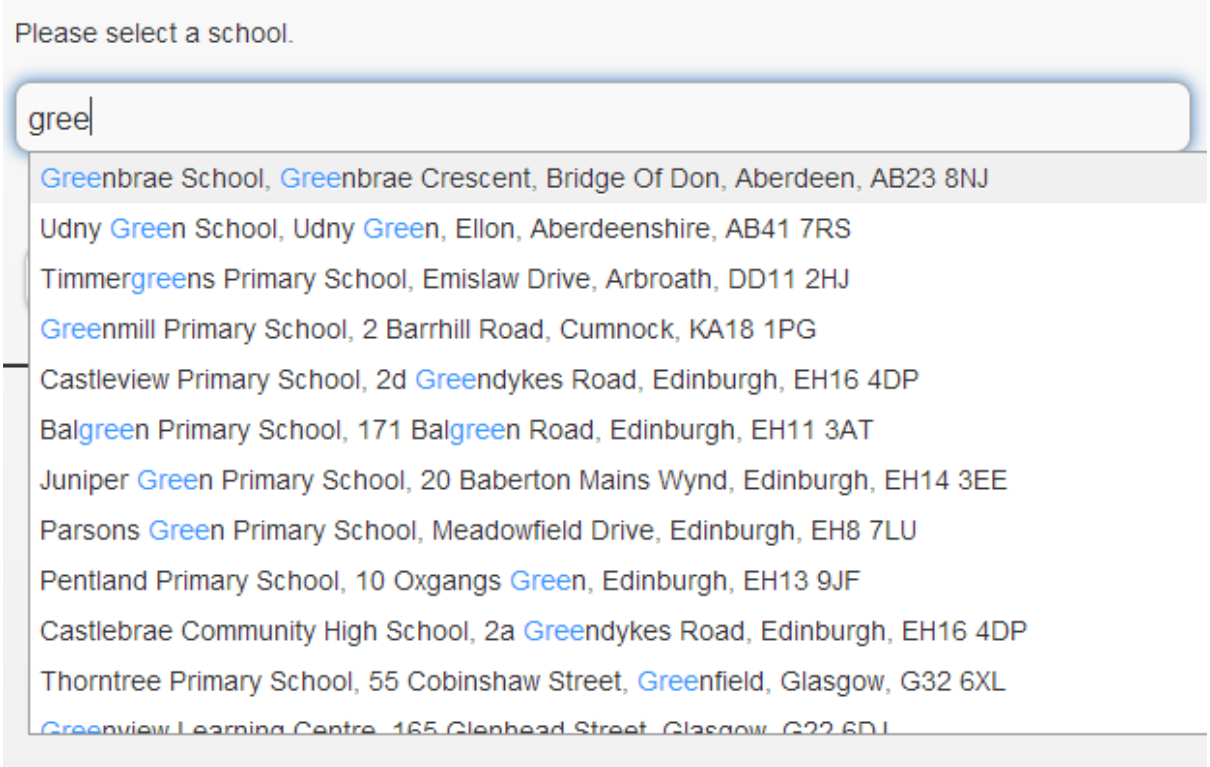

Figure3: Example of the use of a trigram search in a Blaise web questionnaire.

To make this application more portable, the code is written in Classic ASP and uses an Access database. This makes it possible to integrate this application in the .bis package of Blaise and run it without any extra server requirements.

4 ASP Scripts

4.1 Logging to a text file

We modified the page handler script (BiPagHan.asp) so that it logs every form submit in a plain text file. The following items are logged: name of the questionnaire, timestamp (date and time), key value of the respondent, question name and given answer. Such a log allows us to analyze in detail the time interval between given answers and whether respondents tend to go back and forth in the questionnaire.

4.2 Security check

In our institute, respondents normally start a BlaiseIS questionnaire by clicking on a link or posting a form. This link (or form) normally contains the ID of the respondent (KeyValue) as either a GET or POST parameter. This parameter can be manipulated by the respondent. Our simple and robust solution is the addition of a hash value of the KeyValue concatenated with a salt string (normally bigger than 20 characters) in the link (or form). We use the SHA1 algorithm to create this hash. In the BiInterviewerStarter.asp, we added code that recalculates this hash value. Respondents only get a valid interview session when the recalculated hash is identical to the submitted hash. [1]

5 References

[1] Arnaud Wijnant and Edwin de Vet, Performance and Security Enhancements on the Blaise IS standard stylesheet; Proceedings of the 14th International Blaise Users Conference IBUC 2012, p232-236.

[2] Alerk Amin and Arnaud Wijnant, Blaise-On-The-Go: Using Blaise IS with mobile devices; Proceedings of the 14th International Blaise Users Conference IBUC 2012, p237-248 and www.centerdata.nl/link/cmoto.

[3] Arnaud Wijnant and Maurice Martens, C3B: Exploiting the numerous possibilities web technology offers to elevate questions; Proceedings of the 13th International Blaise Users Conference, Baltimore, Maryland USA, October 19-21, 2010, Vol. 13, p.94-105