Binding HTML Elements to Data

To display the data provided by a data source object (DSO), the author binds elements on an HTML page to the DSO. Using the About Data Binding Architecture or the corresponding data binding properties makes it easy. This topic shows how to bind an element to data, lists the elements that support data binding, and describes the capabilities of those elements. Capabilities include support for updating the data to which an element is bound and the data format in which the data is displayed (either HTML or plain text).

Bindable HTML elements fall into two categories: single-valued and tabular consumers. Single-valued consumers bind to a single field of the current record provided by a DSO. Tabular consumers, such as the table element, bind to an entire data set and use their contained elements as a template to repeat the data. The procedure for binding it to data is described later in this article.

This topic contains the following sections.

Binding a Single-Valued Element to Data

The procedure for binding a single-valued element to data is the same regardless of the element. Elements can be bound to data at design time using the DATASRC and DATAFLD attributes, or DHTML Object Model Support for Data Binding using the dataSrc and dataFld properties exposed by the corresponding objects in the Document Object Model (DOM).

Binding a Single-Valued Element to Data at Design Time

Given a text box, for example, a page author can bind that element to data as follows:

Copy

```
<INPUT TYPE=TEXTBOX DATASRC="#dsoComposers"
 DATAFLD="compsr_last">
```

The **dataSrc** attribute in this example specifies the ID, prefixed by a hash mark (#), of a DSO embedded on the page. The hash mark is required. The **dataFld** attribute identifies the field in the data provided by the DSO to which the text box should be bound.

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbsingle.htm

Binding a TABLE to Data

Because the **table** element is a tabular data consumer, it relies on the elements that it contains to bind to the individual fields in the data set provided by the DSO. The contained elements serve as a template, and they are repeated once for each record in the data set. The **table** specifies the **dataSrc** attribute. The contained elements specify the **dataFld** attribute and inherit the **dataSrc** from the table.

When binding a table to data, it is not possible to databind to a tr if it is contained by a tFoot or tHead.

Here's a simple example:

Copy

```
<TABLE DATASRC="#dsoComposer">
<TR>
  <TD><SPAN DATAFLD="compsr_first"></SPAN><
/TD>
</TR>
</TABLE>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbtable.htm

If a databound **table** element includes a tBody element, this is recognized as a part of the template for databinding and is repeated for each row in the dataset, in a manner similar to the **tr** element. Consider the following databound **table**.

Copy

```
<TABLE DATASRC="#dsoComposer">
<TBODY>
<TR>
  <TD><SPAN DATAFLD="compsr_first"></SPAN><
/TD>
</TR>
```

```
</TBODY>
</TABLE>
```

In this example, the table that renders in the browser contains one **tBody** element for each row of data in the data source.

Note When multiple **tBody** elements occur in the document, the tBodies collection can be used in script to reference these objects.

As with single-valued consumers, a tabular data consumer can be bound to data at run time using the DHTML Object Model. The procedure for accomplishing this is described below.

Elements That Support Data Binding

The previous sections use a TEXTBOX, a span element, and a **table** element to illustrate how to bind single-valued and tabular consumers to the data provided by a DSO. Many other HTML elements can be bound to data. Some of these elements support updating the data to which they are bound. Changes can be persisted to the back-end data set if the DSO supports update. Other elements support **About Data Binding Architecture** in addition to the plain text default using the dataFormatAs attribute.

The following sections list additional HTML elements that support data binding.

Read-only Elements

Bindable HTML elements that supply read-only functionality include a, button, div, img, frame, iframe, label, marquee, and span.

Additionally, the **button**, **div**, **label**, **marquee**, and **span** elements support the usage of the **dataFormatAs** attribute or property to render the data to which they are bound as plain text (default) or as HTML. The data displayed by the element is automatically updated as the record pointer maintained by the DSO moves, or as the underlying data to which the element is bound changes. The following are descriptions of these read-only elements.

A

An anchor element applies the data supplied by a DSO to the href attribute; thus, the supplied data should represent a URL. The following is an example of a bound anchor.

Copy

```
<A DATASRC="#dsoLinks" DATAFLD="link_href">
<SPAN DATASRC="#dsoLinks
        DATAFLD="link_friendly"></SPAN></A>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbanchor.htm

BUTTON

A **button** element renders the data supplied by a DSO on its face. The following is an example of a bound **button**.

Copy
```
<BUTTON DATASRC="#dsoLinks" DATAFLD="link_friendly"></BUTTON>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbbutton.htm

DIV

A **div** element is useful for displaying a block of text. The following is an example of a bound **div**.

Copy
```
<DIV DATASRC="#dsoComposer" DATAFLD="compsr_biography"></DIV>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbdiv.htm

FRAME

A **frame** element applies the data supplied by a DSO to its src attribute; thus, the bound value should represent a URL. So that the binding occurs successfully, the DSO supplying data to a bound frame should be present in the head section of the HTML file containing the frameSet element. The following is an example of a bound **frame**.

Copy
```
<HTML>
<HEAD>
<!-- Add DSO reference here -->
</HEAD>
<FRAMESET>
    <FRAME DATASRC="#dsoFAQ" DATAFLD="frame_question" ...>
    <FRAME DATASRC="#dsoFAQ" DATAFLD="frame_answer" ...>
</FRAMESET>
</HTML>
```

The code behind the following button implements a frame-based FAQ. The first frame displays the question, the second frame displays the answer, and the third frame contains navigation buttons. The first and second frames are bound to a two-column table. The first column in the table contains relative paths to pages containing questions. The second column contains relative paths to pages containing the answers. Since the DSO is embedded in the **head** of the outer page containing the **frameSet**, the code behind the navigation buttons drills out of the page in which it resides using the `top` object and references the DSO by its ID. The top object returns an object reference to the outermost containing window.

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbfaq.htm

IFRAME

An **iframe** element applies the data supplied by a DSO to its **src** attribute; thus, the data should represent a URL. In contrast to the **frame** case, the DSO can be declared anywhere on the page. The following is an example of a bound **iframe**.

Copy
```
<IFRAME DATASRC="#DSC1" DATAFLD="iframe_url
">
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbwizfr.htm

IMG

An **img** element applies the data supplied by a DSO to locate, load, and display the image typically pointed to by its **src** attribute. Supplying raw image data through the bound column is not supported. The following is an example of a bound **img** tag.

Copy
```
<IMG DATASRC="#tdcImages" DATAFLD="image"><
/SPAN>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbimg.htm

LABEL

Use a **label** element to describe another bound element on the page. A label applies the data supplied by a DSO to its caption. The following is an example of a bound **label**.

Copy

```
<LABEL FOR=somecontrolid DATASRC="#DSC1" DA
TAFLD="label_col"></LABEL>
```

Since a **label** is associated with other elements indicated by its htmlFor attribute, using a bound **label** within a repeated table can yield unexpected results. If the **htmlFor** attribute references another element within the repeated table, the **label** tag will not be associated with the elements, since there will be multiple elements with the same ID/NAME as a result of the repetition.

LEGEND

The legend element adds a caption to the box drawn by the fieldSet element. When databinding is used with the **legend** element, the data supplied by the DSO appears in the **fieldSet** caption. The following is an example of a bound **legend** element.

Copy

```
<FIELDSET>
<LEGEND DATASRC=#tdcElements DATAFLD="eleme
nt"></LEGEND>
<INPUT CONTENTEDITABLE="FALSE" TYPE="TEXT"
DATASRC=#tdcElements DATAFLD="boundto">
</FIELDSET>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dblegend.htm

MARQUEE

A **marquee** element uses its bound data to replace the text that appears between its opening and closing tags. The following is an example of a bound **marquee**.

Copy

```
<MARQUEE DATASRC="#dsoComposer" DATAFLD="bi
o" DATAFORMATAS="HTML"></MARQUEE>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbmarque.htm

SPAN

Like a **div**, a **span** is a read-only data consumer. Use a **span** to display inline text or limited HTML text. If the **span** is used to display HTML text, that text should not include any HTML block elements. When the current record or the underlying value in the bound column provided by the DSO changes, the **span** reflects the change. The following is an example of a bound **span**.

Copy

```
<SPAN DATASRC="#dsoComposer" DATAFLD="comps
r_last"></SPAN>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbspan.htm

Elements That Support Update

The input (with the exception of the button type), select, textArea, object, and applet elements support updating the data to which they are bound, if the underlying DSO supports update functionality. The following are descriptions of these individual elements.

INPUT

The **input** element represents a set of HTML intrinsic controls. Each of the types that support data binding (input type=checkbox, input type=hidden, input type=password, input type=radio, and input type=text) is detailed in the following sections.

Security Alert Using the **input** element incorrectly can compromise the security of your application. In general, don't assume that user input is valid or benign. Avoid rendering unverified user input as HTML, or transferring that input from an untrusted context to a trusted context within your application. You should review Security Considerations: Dynamic HTML before continuing.

CHECKBOX

Although check boxes allow a value attribute that is used when submitting an HTML form on a page, Windows Internet Explorer MSHTML uses check boxes as simple Boolean selections. Check boxes generate the Boolean values true or false depending on whether they are checked or not. The binding agent coerces the values to and from the underlying data set. The following coercions are supported, based on the type of the bound column.

The following table describes the values that a bound check box expects a DSO to supply for various data types.

Data type	Expected True value	Expected False value
String	"True" | "1" | <nonempty string>	"False" | "0" | <zerolength string>
Integer	nonzero	0
Float	nonzero	0
Date	invalid	invalid
Currency	nonzero	0

The following is an example of a bound check box.

```
<INPUT TYPE=CHECKBOX DATASRC="#dsoSurvey" D
ATAFLD="us_resident"> U.S. Resident
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbchkbox.htm

HIDDEN

The **HIDDEN** type is used to store information in the page that isn't displayed to the user. The element is populated with data from the current record, but it cannot be modified. The following is an example of a bound hidden field.

```
<INPUT TYPE="HIDDEN" DATASRC="#DSC1" DATAFL
D="key">
```

PASSWORD

The **PASSWORD** type is basically the same as the **TEXT** type except that the bound text is not displayed to the user. Be careful when using this input type, because the associated data is fully accessible to scripts through the data-binding object model, as well as directly through the element's value. The following is an example of a bound password.

Security Alert Passwords submitted through INPUT type=password are not encrypted and therefore not secure. Using Secure Hypertext Transfer Protocol (HTTPS) is strongly recommended. You should also use the `post` method for submitting passwords. This submission method hides the password from most users by placing it in the HTTP header. The `get` method submits the password as part of the URL

string, which might be exposed in the **Address** bar and the status bar. For more information, see **Security Considerations: Dynamic HTML**.

```
<INPUT TYPE="PASSWORD" DATASRC="#DSC1" DATA
FLD="password">
```

RADIO

Radio buttons are used to select a single value from a set of alternatives. They can be used to select the value for an enumerated field in a database. One radio button is specified for each of the alternatives using a separate **input**. The name attribute on the **input** determines the logical grouping of alternatives. One value is bound for all the **input**s with the same **name** attribute. All members of a group must specify the corresponding **dataSrc** and **dataFld** attributes. The following is an example of a bound radio button group.

```
<INPUT TYPE="RADIO" NAME="cards" VALUE="mc"

    DATASRC="#dsoOrders" DATAFLD="cardname"
>MasterCard
<INPUT TYPE=RADIO NAME="cards" VALUE="amex"

    DATASRC="#dsoOrders" DATAFLD="cardname"
>American Express
<INPUT TYPE="RADIO" NAME="cards" VALUE="vis
a"
    DATASRC="#dsoOrders" DATAFLD="cardname"
 >Visa
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbradio.htm

TEXT

The **TEXT** type is used as a simple text box. The value of the text box can be bound to a field in the data source using the **dataSrc** and **dataFld** attributes. The following is an example of a bound text box.

```
<INPUT TYPE="TEXT" DATASRC="#DSC1" DATAFLD=
"name">
```

OBJECT

When **dataSrc** and **dataFld** are specified on an **object**, Internet Explorer attempts to bind to the object's default property. The `defaultbind` attribute specified in an object's type

information uniquely identifies the default property. If a default property is not specified in this way, Internet Explorer uses the property with DISPID 0.

The following is an example of binding the default value on the **object** tag.

Copy

```
<OBJECT ID=oControl1 WIDTH=100 HEIGHT=100
    CLASSID="CLSID:xxxxxxxxx-xxxxx-xxxx-xxx
x-xxxxxxxxxxxx"
    DATASRC="#DSC1" DATAFLD="controlData">
</OBJECT>
```

Additionally, Internet Explorer supports binding to arbitrary **object** properties through the object's contained param tags. To do this, apply the **dataSrc** and **dataFld** attributes to the **param** tag. The property's value is initialized with the data supplied by the DSO. The following is an example of binding the **param**s on the **object** tag.

Copy

```
<OBJECT ID="oControl1" WIDTH=100 HEIGHT=100
    CLASSID="CLSID:xxxxxxxxx-xxxxx-xxxx-xxx
x-xxxxxxxxxxxx">
<PARAM NAME="ForeColor" DATASRC="#DSC1" DAT
AFLD="forecolor">
<PARAM NAME="BackColor" DATASRC="#DSC1" DAT
AFLD="backcolor">
</OBJECT>
```

Bindings can be applied simultaneously to the object's default value as well as to its **param** tags.

SELECT

A **select** element supplies the functionality of a drop-down combo box or a list box. Internet Explorer supports binding to a single selected element, but not to to multiple selections.

The items in a **select** control are specified using option tags. The Dynamic HTML (DHTML) Object Model defines an options array that corresponds to the collection of **option** tags for a **select**. Each **option** has a corresponding index, text, and value. **select** has a selectedIndex property that corresponds to the index of the **option** currently selected. If no item is selected, the **selectedIndex** is set to -1. The text attribute of an **option** corresponds to the text that follows the tag and represents the string that is displayed for that **option** tag in the **select** . The value attribute provides the value that is to be returned when the HTML form is submitted. Value is also what is stored into the bound column of the data source.

Initially, the **selectedIndex** property of a bound **select** element will be set to the index of the value in the options array that corresponds to the field of the data source. If the value is not a

member of the options array, the index property is set to -1 and, if the **select** is a combo, no value is displayed. When the user changes the selected item, the corresponding **option** value (attribute) is used to update the value of the bound field of the data source. Validation events are fired as with other controls.

If a **select** element is bound to an empty record, the **selectedIndex** property is not available. This scenario can arise when using the Remote Data Service (RDS) DSO if a **select** element databinds to a blank record. By updating the selected **option**, the blank databound record is updated and RDS returns the modified record to the server.

The following is an example of a bound **select**.

Copy

```
<SELECT DATASRC="#DSC1" DATAFLD="cardname">
<OPTION VALUE=mc>MasterCard
<OPTION VALUE=amex>American Express
<OPTION VALUE=visa>Visa
</SELECT>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbselect.htm

TEXTAREA

A **textArea** is a multirow text box for data input, similar to the **input type=text** intrinsic. As such, it supports the update of the data if the DSO supports the update feature. The following is an example of a bound **textArea**.

Copy

```
<TEXTAREA DATASRC="#dsoComposer" DATAFLD="b
io"></TEXTAREA>
```

Click the button to see a working example.

Code example:
http://samples.msdn.microsoft.com/workshop/samples/author/databind/dbtextar.htm

APPLET

Internet Explorer supports binding to the **param** tags of an **applet**. The following example specifies a binding on the xyz property of the application.

Copy

```
<APPLET CODE=somecode>
    <PARAM NAME="xyz" VALUE="abc" DATASRC="
#Ctrl1" DATAFLD="Column1">
    <PARAM NAME="Title" VALUE="BoundApplet"
```

```
>
</APPLET>
```

Property name resolution

The **name** attribute of the **param** specifies the **basename** of the Java methods used to get and set the value of the property. This is termed **basename** because the method of the Java applet to retrieve the value of the property is `get<`**basename**>. Correspondingly, the method used to set the value of the Java applet is `set<`**basename**>. This naming convention is consistent with the JavaBeans 1.0 specification. The applet used in the preceding section is expected to have methods named getxyz and setxyz.

Notifications

Notifications are not fired by the application when property values change. Property changes are detected only when the current record changes. During this transition, the binding agent interrogates the application and transfers a modified data value, after validation, to the data source. When values are changed directly in the data source, the binding agent is notified by the data source and transfers the value to the application immediately. For more information, see DHTML Event Model Support for Data Binding.

Name space co-mingling

The bound **param** in the preceding example is the most complicated specification possible. That is, it addresses the case where there is a binding to a property specified by the **name** attribute, and there is a parameter to the application with the same **name** with a corresponding **value** attribute. This essentially co-mingles two name spaces: the parameter name space for the application and the binding name space, since the **name** attribute is used for both. In such cases, the application ignores the unexpected **dataSrc** and **dataFld** attributes on the **param** when fetching a parameter (applications explicitly fetch parameters using the getParameter method), and the binding agent recognizes the attributes and attempts binding using the get and set methods specified above.

The simplest example is one where there is no conflict between the parameter name space and the property name space. An example would be:

Copy

```
<APPLET>
    <PARAM NAME="backcolor" DATASRC="#DSC1"
 DATAFLD="Color">
</APPLET>
```

In this example, the backcolor property has only **dataSrc** and **dataFld** attributes; no **value** attribute is specified.

Community Content

Add FAQ