

---

# Using Biferno in XML documents

Teofrasto Bombasto, Parnobis B.V.

## Table of Contents

- 1. The Technique . . . 1
  - 1.1. Embedding in general XML . . . 2
  - 1.2. Embedding in XHTML . . . 3
- 2. Making XML output easy . . . 4
- 3. Conclusions . . . 5
- Bibliography . . . 5

### Abstract

Biferno code can be easily embedded in XML pages. This technique can be used to create dynamic XML documents, i.e. XML documents which are dynamically generated at runtime on the base of an XML template. The template itself is an XML document containing markup and Biferno code “hidden” in processing instructions.

Applications can be found in the design of dynamic sites based on the XHTML standard, in XML publishing, and in the generation of XML documents for electronic data exchange between applications.

## 1. The Technique

Biferno code is usually embedded in HTML pages by enclosing the code between the `<? and ?>` tags. As mentioned in the *Biferno Language Guide* it is also possible to enclose the Biferno code between the `<?biferno and the ?>` tags.

In XML lingo a this is called a *processing instruction* (PI). When an XML parser encounters a PI, a corresponding node type (processing instruction node) is created in the document tree. The node contains the content of the PI.

In our case the node contains Biferno code. The behavior of the XML parser as specified by the XML standard is such that the content of the PI is passed up to the XML application.

In terms of the operation mode of Biferno running along with a Web server, this implies that:

1. An XML document can be written that contains Biferno code in one or more PI's and this document can be validated against its DTD regardless of the embedded Biferno code.
2. The XML document can be deployed on a Biferno-enabled Web site.
3. When a browser requests the XML document, the Web server will pass the XML document as input page to Biferno as usual. Biferno will execute the code contained in the PI's and send the output page back to the server. The server will then send back the output page to the browser.
4. The browser will display the resulting output page, assuming it knows how to display that particular XML document. In practice, four things can happen at this stage:
  - The output page is an XHTML document, which the browser is able to display.
  - The output page is a general XML document, and the input document specifies an XSL stylesheet implementing a transformation to (X)HTML. The browser will be able to display the resulting (X)HTML document.

- The output page is a general XML document, and the browser either can *not* access the XSL stylesheet defining a transformation to (X)HTML or can access the XSL stylesheet defining a transformation into general XML. In both cases the browser will be required to display a general XML document, which is supported by Internet Explorer versions 5.x and 6.x.<sup>1</sup>

Limited support for this feature should not really worry us, because the most likely use of dynamic generation of XML documents via Biferno is for application use, not for human consumption.

- The output page is either non valid or non well-formed XML. Notice that this can happen even if the input page was well-formed or valid, because the execution of the Biferno code may produce a non valid or non well-formed document.

Let's now see a couple of examples.

## 1.1. Embedding in general XML

Assume we would like to fetch user data (user id, description, a number to send SMS messages, etc.) from a relational database and format it as a simple XML document containing a list of users. The following document contains embedded Biferno code that implements a query to the database via ODBC, fetches the user data from the database and returns data according to the tags specified in the DTD:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE userlist [
<!ELEMENT userlist (user)* >
<!ELEMENT user (id, description, email, sms)+ >
]>

<?biferno
dbconn = db("DSN=myDB;UID=sa;PWD=myPwd", "odbc")
query = "SELECT * FROM demo.dbo.USER_users"
dbconn.Exec(query)
nrec = dbconn.GetCurRecs()
?>
<userlist>
<?biferno
for (i = 1; i <= nrec; i++)
{
    recArray = dbconn.FetchRec()
    $('<user>'
    $('<id>'+recArray["USER_ID"]+'</id>'
    $('<description>'+recArray["USER_DESCRIZIONE"]+'</description>'
    $('<email>'+recArray["USER_E_MAIL"]+'</email>'
    $('<sms>'+recArray["USER_SMS"]+'</sms>'
    $('</user>'
}
?>
</userlist>
```

When this page is deployed to a server and then requested via a browser, the result is similar to the following:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE userlist [
<!ELEMENT userlist (user)* >
<!ELEMENT user (id, description, email, sms)+ >
]>
<userlist>
  <user>
    with each element, your message may vary.
```

```

    <id>001</id>
    <description>John Smith</description>
    <email>jsmith@myorg.com</email>
    <sms></sms>
  </user>
  <user>
    <id>002</id>
    <description>Adam Thefirst</description>
    <email>atf@myorg.com</email>
    <sms>+336728891</sms>
  </user>
</userlist>

```

Since the XML output page is not XHTML, no stylesheet is specified, and the requesting application is a browser, the browser will attempt to show the “raw” XML. Much more interesting is the case in which the requesting application is not a browser and will further process the output XML document. You can experiment with command-line requests to the server using the `wget` utility [<http://www.gnu.org/software/wget/wget.html>]. The usage is as follows:

```
wget -q -O - http://<your_site>/<your_page>
```

## 1.2. Embedding in XHTML

The following code shows the same Biferno code as above, now embedded in a XHTML document and resulting in another XHTML document:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >
<html>
  <head>
    <title>Embedding Biferno code in XHTML</title>
  </head>
  <body>
    <div>
      <b>Biferno code embedded in an XHTML page</b>
      <hr/>
      <?biferno
dbconn = db("DSN=myDB;UID=sa;PWD=myPwd", "odbc")
query = "SELECT * FROM demo.dbo.USER_utenti"
dbconn.Exec(query)
nrec = dbconn.GetCurRecs()
?>
      <?biferno
if (nrec)
{
  $'<table>'
  for (i = 1; i <= nrec; i++)
  {
    recArray = dbconn.FetchRec()
    $'<tr>'
    $'<td>'+recArray["USER_ID"]+'</td>'
    $'<td>'+recArray["USER_DESCRIZIONE"]+'</td>'
    $'<td>'+recArray["USER_E_MAIL"]+'</td>'
    $'<td>'+recArray["USER_SMS"]+'</td>'
    $'</tr>'
  }
  $'</table>'
}
?>
    </div>

```

```
</body>
</html>
```

When this page is deployed to a server and then requested, the result is similar to the following:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" >

<html>
<head>
  <title>Embedding Biferno code in XHTML</title>
</head>
<body>
  <div>
    <b>Biferno code embedded in an XHTML page</b>
    <hr>

    <table>
      <tr>
        <td>001</td>

        <td>John Smith</td>

        <td>jsmith@myorg.com</td>

        <td></td>
      </tr>

      <tr>
        <td>002</td>

        <td>Adam Thefirst</td>

        <td>atf@myorg.com</td>

        <td>+336728891</td>
      </tr>

    </table>
  </div>
</body>
</html>
```

## 2. Making XML output easy

A simple Biferno function makes the output of XML elements from Biferno code easier and less error-prone:

```
function string PrintElement (string tag, string content, ...)
// pass tag, content, plus optional attribute_name attribute_value pairs
{
  str = '<' + tag
  tot_val = curScript.GetTotVariables()
  for (i = 3; i < tot_val; i = i + 2)
  {
    str += ' ' + curScript.GetIndVariable(i) + '=' + curScript.GetIndVariable(i+1)
  }
  if (content.Zap())
  {str += '>' + content + '</' + tag + '>'}
  else
  {str += '/>'}
}
```

```
return str  
}
```

Use as follows:

- To build the string `<mytag>and its content</mytag>`, call `PrintElement('mytag', 'and its content')`.
- To build the string `<mytag a="x" b="y">and its content</mytag>`, call `PrintElement('mytag', 'and its content', 'a', 'x', 'b', 'y')`.
- To build the string `<mytag/>`, call `PrintElement('mytag', '')`.

## 3. Conclusions

Biferno's usage domain is not limited to dynamic generation of HTML pages, but is readily extended to dynamic generation of XML documents. On one hand, this can be used to build XHTML-compliant web sites, which has a wealth of advantages.

We believe that the true value of Biferno will be unleashed in its use in the dynamic generation of XML documents for application interchange.<sup>2</sup>

## Bibliography

Biferno Language Guide Tabasoft S.a.s.

---

<sup>2</sup>It is worth mentioning that, at present, the use of Biferno as described here will not work for XML documents using multi-byte character encodings (e.g. UTF-16, ISO-10646-UCS-4). The main reasons are that the Biferno code parser and the string class do not currently support multi-byte encodings.