Handling Binary Data in XML

Wahyuning Diah
weezery2002@yahoo.com.au

Introduction

XML is eXtensible Markup Language that allows the users to define their own tags [1]. It is extensible because it is not like another markup language such as HTML (Hypertext Markup Language) that has predefined tags. As in XML, it is the process of creating the tags for markup language that can be different from one application to another application.

In XML, the use and the structure of the tags are related to users' point of view. The users will define how the data are collected and arranged in a hierarchical order. Therefore, the users can choose the tags as the markup that can be rich in information (semantic content) since the tags indicate the content of information rather than the way of displaying the information. The following is the example of the tags in XML and its content:

<customer>

<name>Diah<name>

<preference>books<preference>

<customer>

Between the tags (name and preference), the XML document holds the data (Diah and books) for each particular element.

There are two types of data that can be included as the content in the XML document, plain text (ASCII text) as a parsed XML and binary data as an unparsed XML [2]. In the general computing term, binary data refers to data that can be represented in a series of ones and zeros. It is usually called the digital data because the binary data has two elements, one and zero, which are the same with on and off state in the electric pulse. In relation to the content of the XML document, the term binary data is used to represent bitmap (picture) data files, sound files and movie files [2]. For example, an XML document can consist of the customer signature, the picture of product, or even the chunk of a song/movie of a particular artist.

There are some methods that can be implemented to handle binary data in XML such as [3,4]: are representing the binary data by means of external entity, representing binary data using MIME data types, embedding binary data in CDATA section and notation and representing the binary data using ID attribute. Now, there is a simple way to directly reference an external binary data type that the XML uses. This application is called SMIL, the Synchronized Multimedia Integration Language [5].

This paper discusses only two ways on how to include or attach binary data into the content of the XML document and also to retrieve it to be displayed in a certain media such as web page. The two methods discussed are representing the binary data by means of external entity and notation and representing the binary data using ID attribute.

Attaching Binary Data into XML and Extracting/Retrieving Binary Data from XML

There are two ways that can be used to attach binary data into XML and extract or retrieve binary data from XML:

1. **Represent the binary data by means of external entity and notation**

   This method uses the ENTITY type and specifies a declared NOTATION included in DTD (Data Type Definition) structure. NOTATION is needed to tell the XML parser that there is an unparsed XML data attached to the XML document. The way the method works is to reference the binary data (i. e picture) from within a document as the value of an attribute (as the img element and src attribute in HTML tags).

   For example, in the following document, the flowerPicture entity is declared to have a GIF notation, and used as the value of the empty flowerDisplay element's pic attribute.

```
<?xml version="1.0"?>
```

```
<!DOCTYPE collection SYSTEM "flower.dtd">
<collection>
<flower>
<flowerName>Roses</flowerName>
<flowerDescription>red roses</flowerDescription>
<flowerPrice>&#xA3; 6</flowerPrice>
<flowerDisplay pic="flowerPicture"></flowerDisplay>
</flower>
</collection>
```

The DTD that includes ENTITY and NOTATION for the XML document above can be seen below [6]:

```
<?xml version="1.0"?>
<!NOTATION gif89a PUBLIC "-//Compuserve//NOTATION Graphic Interchange Format89a//EN" "gif">
<!ENTITY flowerPicture SYSTEM "rose.gif" NDATA gif89a>

<!ELEMENT collection (flower)+>
<!ELEMENT flower (flowerName, flowerDescription, flowerPrice, flowerDisplay)>
<!ELEMENT flowerName (#PCDATA)>
<!ELEMENT flowerDescription (#PCDATA)>
<!ELEMENT flowerPrice (#PCDATA)>
<!ELEMENT flowerDisplay (#PCDATA)>

<!--Attributes-->

<!ATTLIST flowerDisplay
        pic ENTITY #REQUIRED
>
```

The declaration above shows the relation between the flowerDisplay element and its attribute pic in XML document to connect with the image file (GIF format in this case) that specified in DTD file using the flowerPicture ENTITY.

In order to convert the pic image in flowerDisplay element so that the rose.gif can be included in the XML document, there is a need for another application software (let say Java) to do it. The following piece of codes does that [3]:

```
String cStrTempElementName;
StringBuffer cSBufferTempElement = new StringBuffer();
private Hashtable m_cHtableNotation;
private Hashtable m_cHtableEntity;

if(cStrTempElementName.equals("flowerDisplay"))
{
    if(m_cHtableEntity.containsKey(attrs.getValue(0)))
   {
```

```
        Vector cVectorTempEntity = (Vector)m_cHtableEntity.get(attrs.getValue(0));
        cSBufferTempElement.append("mode="'+cVectorTempEntity.elementAt(1).toString()+'"");
        cSBufferTempElement.append("type="'+(String)m_cHtableNotation.get
                        (cVectorTempEntity.elementAt(1).toString())+'">");
                        cSBufferTempElement.append(cVectorTempEntity.elementAt(0).toString());
        System.out.println("\n\n flowerPicture is found in the External Entity List");
    }
}
```

The first if statement is about to compare whether the element name is equal to flowerDisplay. The next lines of the code are trying to retrieve the value of the attribute and then assign and print it to notify that the flowerPicture ENTITY is found.

After the conversion, the XML document result will be as the following:

```
<?xml version="1.0"?>
<collection>
<flower>
<flowerName>Roses</flowerName>
<flowerDescription>red roses</flowerDescription>
<flowerPrice>&#xA3; 6</flowerPrice>
<flowerDisplay mode="picture" type="gif "> rose.gif
</flowerDisplay>
</flower>
</collection>
```

The flowerDisplay is now carrying a rose.gif file on its content and there are two attributes which are mode and type.

2. Represent the binary data using ID attribute

This method works by using an ID attribute on the flowerDisplay element [4]. Each ID attribute in the collection will be given the numeric value “1” and so on. So there is a slightly changing on the XML document and the DTD structure.

In DTD structure, the ENTITY and NOTATION declarations are removed and also the flowerDisplay element is changed into EMPTY element, which holds the ID attribute. The ID attribute is like the “href” mechanism used in HTML to link the flowerDisplay element directly to the image file containing the required image.

```
<?xml version="1.0"?>
<!DOCTYPE collection SYSTEM "flower.dtd">
<collection>
    <flower>
        <flowerName>Roses</flowerName>
        <flowerDescription>red roses</flowerDescription>
        <flowerPrice>&#xA3; 6</flowerPrice>
        <flowerDisplay ID="1"></flowerDisplay>
    </flower>
</collection>
```

The DTD for the XML document above can be seen below:

```
<?xml version="1.0"?>

<!ELEMENT collection (flower)+>
<!ELEMENT flower (flowerName, flowerDescription, flowerPrice, flowerDisplay)>
<!ELEMENT flowerName (#PCDATA)>
<!ELEMENT flowerDescription (#PCDATA)>
<!ELEMENT flowerPrice (#PCDATA)>
<!ELEMENT flowerDisplay #EMPTY>

<!--Attributes-->

<!ATTLIST flowerDisplay
                ID  #REQUIRED
>
```

The way to reference the flowerDisplay in the XML document and its associated number in the CSS (Cascading Style Sheets) are by written flowerDisplay#1 (with the pound sign to assign the ID attribute used). The background formatting text is used on the declaration to hold the value of the image. As for the other formatting texts on the flowerDisplay are purposed to make the appearance of the picture nicer. For example, by specifying the position of the image (left, top, relative, float) and also the image size (width, height), its border, and its display (starts at the beginning of a line).

The CSS document for the XML document above can be shown below:

```
flowerDisplay#1
{
        display: block;
    background:url(rose.gif);
        border: solid green;
        float:left;
        width:80px;
        height:90px;
        position:relative;
        left:0px;
        top:-30px;

}
```

In Shankar's [3] opinion, the first method is the elegant and efficient way of representing the binary data. It is because the data is modeled in the XML document and the details of the binary data (source, filename) are abstracted in the DTD. This is pretty much different from the second method, which only specify the general DTD file as for the image file is represented in Cascading Style Sheets for the layout formatting. The second method will be more difficult to implement if the XML document has a lot of image files.

But the advantage of the second method is that it is easy to implement since there is no need for a converter application (such as Java) that parses the XML file and the DTD to gather information about external entity references.

The two methods are different in term of the XML documents that will be handled. The first method is suitable for the big XML documents and the second method is for small XML documents.

Conclusion

Binary data can be categorized as unparsed data in XML. The examples of the binary data are bitmap (picture) data files, sound files and movie files. As an unparsed data, therefore binary data needs to be handled in a special way. There are two approaches that are discussed in this paper, representing the binary data by means of external entity and notation and representing the binary data using ID attribute.

The first method works by including ENTITY type and specifies a declared NOTATION included in DTD (Data Type Definition) structure. Another application, Java for example, is needed to convert the file included in DTD into XML document.

The second method uses ID attribute on the element that the image attached to. Therefore there is a link between the XML document and the CSS by number of the ID. In order to retrieve the image to display in the web page, CSS includes the file name of the image.

The first method is suitable for the big XML documents and the second method is for small XML documents.

References

1. Deitel, H.M., Deitel, P.J, Nieto, T.R., Lin, T.M. & Sadhu, P. (2001), "Creating Markup with XML", in *XML How to Program*, Prentice-Hall, New Jersey, pp. 112-132.

2. "XML FAQ - Using entities for images". Available: http://www.ucc.ie:8080/cocoon/xmlfaq [Accessed May 3, 2003].

3. Shankar, G. (February 2002), "Embed binary data in XML documents three ways". Available: http://www-106.ibm.com/developerworks/xml/library/xbinary/index.html?open&l=136,t=gr,p=xb2b [Accessed May 7,2003].

4. Casey, D. (April 2003), "Topic 14: Cascading Style Sheets – Part 6. Images". CPE5011 Internet Application Development Handout.

5. Rein, L. (24 July 1998), "Handling Binary Data in XML Documents". Available http://www.xml.com/pub/a/98/07/binary/binary.html [Accessed May 2, 2003].

6. Indrawan, M. (2003), "DTD - Notations non XML data". CSE4500 Information Retrieval Handout.