Dynamic Modification of XML Documents: External Application Invocation from XML

Ibrahim Cingil
and
Asuman Dogac
and
Ender Sevinc
and
Ahmet Cosar
Middle East Technical University

In this paper we describe a mechanism for invoking external applications from XML documents and returning the results back into the documents such that it becomes possible to create dynamic XML documents with content fetched from external resources such as remote databases or legacy applications. One possible application is to support dynamic electronic catalogs in XML where the catalog content is automatically and dynamically updated to contain the most recent data external to the document by accessing resources like legacy applications or databases each time it is parsed.

The mechanism introduced involves a way of specifying applications to be invoked from an XML document through a new processing instruction, called EXECUTE. The proposed approach necessitates a "variable" mechanism to be able to pass input parameters to the invoked applications and to use the returned results. A variable mechanism is not readily available in XML therefore we have modified the code of an XML parser such that it became possible to redefine the values of XML entities and thus to use them as variables. XML-RPC from UserLand is used for remote method invocation.

When it comes to integrating the returned results into the original document, as long as a single value is returned, it can be readily inserted into an XML document. However a piece of XML document may also be the result of the external application. In this case if a validating parser is being used, as in our case, then the DTD specified should be that of the final document.

The paper concludes by discussing the future work.

Address: Software Research and Development Center
Middle East Technical University
06531 Ankara Turkiye
asuman@srdc.metu.edu.tr

1. INTRODUCTION

XML *(http://www.w3.org/TR/REC-xml-19980210)* is the standard for data exchange over the Internet. XML data is self describing and therefore it makes it possible for programs to interpret this data. Hence it enhances the ability of remote applications to interpret and operate on documents fetched over the Internet. However with the increasing use of XML it becomes important for several applications to interact with other sources from an XML document to dynamically create some XML data.

As an example assume an electronic catalog defined in XML where one of the elements describing a product is the available stock which resides in the stock inventory database. Obviously for this item to dynamically reflect the changes in the related database, the "in stock quantity" must be accessed from the database each time the electronic catalog is accessed.

As another example consider creating an integrated electronic catalog customized according to the needs and preferences of a particular user where the information needs to be obtained from several databases and/or legacy applications. For each of these systems wrappers can be developed to produce the required data in XML format. Then by properly invoking these wrapper programs from the customized catalog document in XML, an electronic catalog can be constructed dynamically and automatically which reflects the most recent changes in the data.

In this paper we describe such a mechanism for dynamic modification of XML documents by returning the results of externally invoked applications into the documents.

The advantage of this approach over dynamic generation of XML documents is as follows: our approach makes it possible to specify where to get the data and how to integrate it in a document. XML documents dynamically generated from a number of possibly heterogeneous resources can thus be flexibly integrated. When a change becomes necessary in the document, for example to invoke another resource, this can be accomplished by just changing the document. A server program developed to do this task on the other hand requires the modification of the code, each time a change is necessary. Furthermore where to get the data is hard coded in a server program.

2. IMPLEMENTATION

In introducing a mechanism for dynamic modification of XML documents there are three issues that need to be addressed: a way of specifying application invocation, a mechanism for remote method invocation, a mechanism for plugging in the returned result to the XML document.

2.1 A way of specifying application invocation

In XML, the NOTATION facility is used for invoking external applications. However NOTATION facility is one direction, that is, it is possible to pass input parameters to the external applications, however there is no mechanism to retrieve the result of the invoked application back into XML. Therefore we have introduced a new processing instruction to XML which we call "EXECUTE". In the following we provide the template of the proposed processing instruction:

```
<?EXECUTE type ="application_type" name ="application_name"
source="http://www.srdc.metu.edu.tr/sc/R1/application_to_be_invoked"
userid="userName" password="Password"
input=[<call.parm name="customer.name"> &input_param; </call.parm>]
output=[<call.parm name="customer.id"> &output_param; </call.parm>] ?>
```

The EXECUTE statement can appear anywhere parsed data is allowed. The XML parser passes the execution to the processing application. The "type" attribute specifies the type of an application like a legacy application, an SQL, or PL/SQL query, or an XML-QL query. New application types can be added as needed. The "name" attribute specifies the name of the application to be executed and the "source" attribute either provides a URL or the source itself is written in between brackets. The input parameters of the process, if any, can be given by the "input" attribute which are evaluated by the processing application. Note that general XML entities are allowed inside the input parameters. Alternatively, the input attribute can specify a URL where the input parameters can be found. Similarly the output parameters of the process, if any, are specified by the "output" attribute which specifies a URL indicating where the output will be stored. The output attribute alternatively can give a variable name to store the output values. Clearly there is a need to use "variables" in the definition of the "EXECUTE" processing instruction to store output values. Variables are not readily available in XML. Furthermore DOM Level 1 *(http://www.w3.org/ TR/REC-DOM-Level-1)* parsers do not allow the value of the entity to be changed. However variables can be accommodated into XML with some changes in the parser to allow the XML's <!ENTITY ..> definitions to redefine a previously defined entity. This is the approach taken in this implementation where the code of IBM's XML4J Parser *(http://www.alphaworks.ibm.com)* has been modified such that it became possible to redefine the value of an entity or even to create a previously undefined entity which are then used as variables.

2.2 A mechanism for remote method invocation

In the work described in this paper, XML-RPC *(http://helma.at/hannes/xmlrpc/)* protocol developed by UserLand *(http://www.userland.com)* is used. XML-RPC is a remote protocol that uses XML to encode procedure calls and responses. An XML-RPC message is an HTTP-POST request. The body of the request is in XML. A procedure executes on the server and the value it returns is also formatted as XML. The XML-RPC library can be embedded into any Web server framework that supports reading HTTP POSTs from an InputStream. On the server side, one can either embed the XML-RPC library into an existing server framework, or use the built-in special purpose HTTP server. At the client side the user prepares the parameters for his RPC request and then sends those parameters to the server and waits for the response. Both the source code and the details of implementation are available from *http://helma.at/hannes/xmlrpc/*.

2.3 A mechanism for plugging in the returned result to the XML document

It should be noted that when the result of an external application is inserted into the XML document, the document should still conform to its corresponding DTD if a validating parser is being used. As long as single value is returned, this does not constitute a problem. If the returned result is a piece of an XML document, the DTD should be that of the final integrated document. Also the wrapper application must provide the proper tags for the data it creates.

3. EXAMPLES ON USAGE

In this section the benefits of the technique described is clarified through examples. In the first example the value of an XML element is retrieved from a database into an electronic catalog expressed in XML. Consider the following electronic catalog DTD:

```
<?xml version="1.0" encoding ="UTF-8"?>
<!DOCTYPE electronic.catalog [
<!ELEMENT catalog (catalog.entry*) >
<!ELEMENT catalog.entry (product+) >
<!ELEMENT product (product.id, price, in.stock.quantity) >
<!ATTLIST product ident CDATA #REQUIRED>
<!ELEMENT product.id (#PCDATA) >
<!ELEMENT price (#PCDATA) >
<!ELEMENT in.stock.quantity (#PCDATA) > ]>
```

The following is the XML document conforming to this DTD that receives the value of "in stock quantity" from an Oracle database:

```
<catalog> <catalog.entry> <product ident="MB333-64">
<product.id> "MotherBoard 333-64" </ product.id>
<price> 15 </price>
<in.stock.quantity>
<?EXECUTE type="database application; Oracle;SQL"
name=" retrieve.in.stock.quantity "
conn_str="jdbc:oracle:thin:@144.122.230.16:1521:orcl"
username="system" password="manager"
source=[" SELECT available_quantity_attribute
        FROM stock_table
        WHERE product_id_attribute = product_id "]
input=[<call.parm name="id"> "MotherBoard 333-64" </call.parm>]
output=[<call.parm name="name"> &in.s.quantity;</call.parm>] ?>
&in.s.quantity;
</in.stock.quantity> </product> ... </catalog.entry> ... </catalog>
```

Note that, "avail_quantity_attribute" and "product_id_attribute" are the attributes of the stock_table in the database whereas "in.s.quantity" is an XML entity used as

a variable. The query result is stored in "in.s.quantity" as the output variable and used in the XML document as an entity. Each time the "<in.stock.quantity>" is parsed, the query is executed to retrieve the current stock quantity for the product in question. In this example the value of the product.id, "MotherBoard 333-64", is hard coded to the processing instruction; however our implementation also allows for passing entity values as input parameters to the processing instruction. Such a need may arise, for example, when it is necessary to pass the output value of a processing instruction as an input to one of the processing instructions that follows.

As another example consider the creation of an integrated electronic catalog. Now assume that each product information is stored on a different resource. A customized catalog according to the needs and preferences of a particular user can be obtained by the use of the EXECUTE statement, an example of which is shown in the following:

<catalog> <catalog.entry> <product ident="MB333-64">
<?EXECUTE type="Unix_Shell" name=" Get.ProductInfo "
username="system" password="manager"
source=[" http://www.srdc.metu.edu.tr/sc/wrappers/Get.ProductInfo"]
input=[<call.parm name="id"> "MotherBoard 333-64" </call.parm>]
output=[<call.parm name="name"> &product;</call.parm>] ?>
&product; </product> <product ident="B 2358">
<?EXECUTE type="Unix_Shell"
name=" Get.ProductDetails "
username="system" password="manager"
source=[" http://www.ebooks.com/sc/wrappers/Get.ProductDetails"]
input=[<call.parm name="id"> "eBoys" </call.parm>]
output=[<call.parm name="name"> &product;</call.parm>] ?>
&product; </product> ... </catalog.entry> ... </catalog>

Note that a wrapper program for each of the resources (let it be a legacy application or a database) involved produces an XML document as presented in the following:

<product ident="MB333-64">
<product.id > "MotherBoard 333-64" </ product.id>
<price> 15 </price>
<in.stock.quantity> 20 </in.stock.quantity> </product>

The resulting catalog is as follows:

<catalog> <catalog.entry> <product ident="MB333-64">
<product.id > "MotherBoard 333-64" </ product.id>
<price> 15 </price>
<in.stock.quantity> 20 </in.stock.quantity>
</product>
<product ident="B 2358">
<product.id > "eBoys" </ product.id>
<price> 18 </price>

<in.stock.quantity> 0 </in.stock.quantity>
</product> ... </catalog.entry>... </catalog>

As this example clearly demonstrates, our approach provides a way of describing and integrating dynamically created XML documents from different heterogeneous resources.

4. DISCUSSIONS AND FUTURE WORK

In this paper we have introduced a processing instruction to XML for invoking external applications so that their results can be integrated into XML documents. For this purpose IBMs XML4J Parser has been modified and XML-RPC has been used in order to invoke remote applications transparently and pass parameters and results. The benefits of the mechanism introduced are demonstrated through simple examples. An alternative to introducing a processing instruction could be modifying NOTATION facility. As previously mentioned, NOTATION facility in XML is used for invoking external applications but it does not involve any mechanism to return values. However it can be extended to accommodate the results of an external application it initiates. An external application may be required to generate its results in XML so that it can readily be integrated into XML data. Finally, as identified in the work described, a variable mechanism is necessary for XML. We have used XML entity definitions for this purpose. Alternatively, a more general purpose variable structure can also be added to XML.

The code of the system developed is available from *http://www.srdc.metu.edu.tr.*