Internet Client Graphics Generation Using XML Formats

Javier Rodeiro and Gabriel Pérez

Depto. Informática ,
Escuela Superior Ingeniería Informática,
Edificio Politécnico, Campus As Lagoas s/n
Universidad de Vigo,
32004 OURENSE, Spain
jrodeiro@uvigo.es
http://www.ei.uvigo.es/~jrodeiro/

Abstract. For a long time abstractions have been commonly used for representing graphical elements. The use of network systems (in particular INTERNET) for this type of elements has the difficulty of trying to submit the enormous volume of information contained. One possible solution to this could be the transference of an abstracted definition of the charts and to perform the drawing in the client which is requesting the graphical form. Previous approaches have been based either on SVG [1] or XML [2] but they have the inconvenient of being unsuitable for future incremental developments. In the work presented here a new approach based on XML and Java is proposed. It does provide not only a working tool but it is also an open and incremental system regarding the possible techniques which can be applied during the drawing process in the client and during the transference of information as a XML document.

1 Introduction

When data are registered in any kind of source (electronic, paper, etc.) we can foresee that any type of consult of the stored information is intended. In order to make those data more accessible and to facilitate operations with them there are a wide range of informatics' applications, from the simplest worksheet to the most complex database management systems.

Once this friendly and efficient system is created, problems arise in relation to enable data presentations in such ways which are complete and easy to understand and that they do not lose their reliability. To achieve this there are graphics and data diagrams.

One of the common difficulties when performing queries to a remote data base or worksheet to obtain a graphic result (i.e. as a reliable image which represents the consulted data) occurs when trying to send the resulting chart through the network. The main problem is that they are big in size. Reductions in the size for a better handling inside the network can be sorted out with file compressors (generally based in losses of information) or by using vector graphics [1].

In this project we intend to solve this problem by reducing the data fluxes during the communication between the client and the server when representing the results. To achieve this, instead of submitting the final document as JPEG, BMP, etc., the server will send a graphic representation as XML (eXtensible Marked Language), through an applet Java, to the client. This has been defined previously to generate the graphic from the consulted database.

2 Proposed project

2.1 Description

Graphic description was performed in XML, by means of the following DTD (Data Type Definition) [3].

```
<!ELEMENT grafico (angular | circular)>
 <!ELEMENT angular(nombre,ymax,nelementos,distancia,ancho,barra*)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ymax (#PCDATA)>
  <!ELEMENT nelementos (#PCDATA)>
  <!ELEMENT distancia (#PCDATA)>
  <!ELEMENT ancho (#PCDATA)>
  <!ELEMENT barra EMPTY>
   <!ATTLIST barra
     valor CDATA #REQUIRED
     indice CDATA #REQUIRED>
```

Implementation was carried out by using a Java platform, for both XML documents generation and rendering, originated from the XML definition. In this implementation it is remarkable the independence of this platform, between the server and the clients. For this reason the use of applets was chosen due to the fact that they can be executed within a web navigating environment and only the address (URL) of the XML document is needed. Thus the described method provides to the client clear information about the server which has processed and sent the data.

The communication between the client and the server is as follows:

1. The client (for example from a website) logs into the server and requests one chart based on certain data and specifies the graphic type and the format desired.
2. Based on this request the server originates a XML file which contains the components definitions of the image. At the same time a web page is created in which the correspondent applet to the required graph is included.
3. This applet creates a communication channel with the initial document, then extracts the information contained and represents this in the navigator.

2.2 Interpreting XML

When handling XML documents it is of great importance to follow the definition indicated in the DTD which corresponds to the type of documents under use. Within the client-server structure, both elements directly operate in XML. In addition it should be taken into account that they are completely independent and therefore because of the lack of direct communication between them it is essential to ensure that they both work on the same XML base.

Regarding the server component, we are currently implementing reference servers in order to establish the foundations of later versions which will allow to work with different information sources (databases, keyboard directly, from text files...). It is also noticeable that this system has been built by using an open methodology, so that this project can be easily improved by third persons in relation to either rendering methods or to the XML structure of the representation.

The API Java for XML and JAXP, provides great advantages in the development of these servers and, indeed the necessary methods to originate the result contained in XML documents are common to all servers.

However, in relation to the client, this is not the case mainly due to the restrictions in size imposed to the applets (they must be around 5KB, including the subclass gdXML, which has been employed here and it is described below). In order to reduce the applets size we have focused the development in following two premises: firstly, the subclasses gdXML should implement their own XML interpreter for their correspondent DTD; and secondly, the structure of the file containing the XML document must be as much simple as possible (i.e. it should not allow line breaks in the label space, for example).

When performing the implementation the task of building a XML interpreter in every subclass created increases slightly the effort but allows to reduce the resulted coding substantially.

Additionally, preparing the client to work with a simple structure in the archive also influences in the final size. This is achieved by increasing the complexity at the server level, since that it is in charge of creating the XML document in the most simple way.

2.3 Client

Initially one single applet with the function of interpreting and representing all chart types was used (gdXMLapp). In order to do this we employed the class renderer which creates an URL object for the XML file, opens the flux of information to it and represents the final output in the navigator.

The main problem with this procedure was the size of the class renderer which increased enormously. This increment in size was due to the implementation of the necessary methods to represent the new graphic solutions. In addition to this, more procedures should be added for the applet to call the class renderer and to indicate the type of graphic to be used. In view of this we decided to change the classes hierarchy so that from the mother class, gdXML (which implements all the necessary methods to interpret the XML document) the classes representing the graphics are subsequently developed.

The class gdXML includes abstract methods [1] for drawing procedures. This allows the generation of new subclasses following a pre-established format, which facilitates the programming of both the applets and these new subclasses included in the formers (Fig. 1).

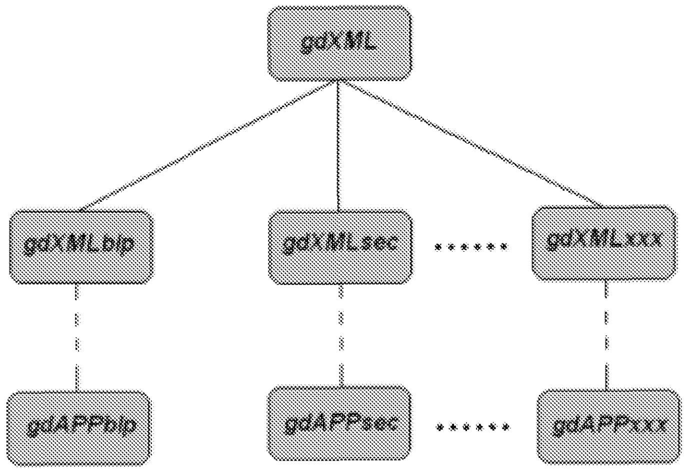

Fig. 1. Structure of the client classes hierarchy

The abstracts methods listed below are proposed:

rname(Graphics g) Chart name.
rLegend(Graphics g) It places the legend within the applet environment.
rElements(Graphics g) It draws the elements which conform the graphic (columns, sectors,. . .).

To date two classes have been implemented(including the corresponding applets), which were gdXMLblp (creation of columns, lines and scatter charts) and gdXMLsec (pie charts). They can be described as an specialisation of the class renderer because each one is able to interpret a specific chart.

The main advantage provided here is to reduce the final size of the file as a consequence of including only the minimum number of drawing functions to represent the requested chart. In addition to this, the implementation effort is also reduced by inheriting from the class gdXML the necessary methods for interpretation purposes.

These applets work as follow:

[1] These methods were not implemented in gdXML, instead each subclass created from it is in charge of its own implementation

1. The applet receives the name of the file to be interpreted from the web page code.
2. The applet creates a flux from the URL of the file.
3. The applet runs the class builder procedure and uses this flux as parameter.
4. The builder procedure obtains the necessary values to build the chart.
5. The applet uses the necessary visualization methods to create the resulting image in the navigator (Fig. 2).

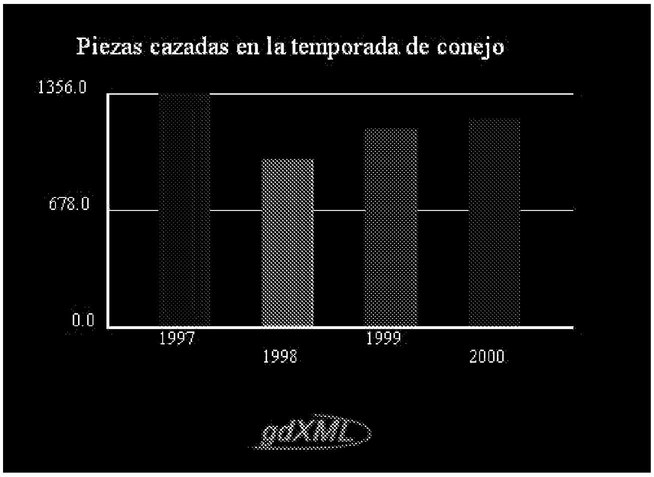

Fig. 2. Example of the resulting image in the navigator

2.4 The applets gdAPPblp and gdAPPsec

The applet gdAPPsec (through the class gdXMLsec) represents the pie charts and separates the different sectors which constitute the total figure. To do this the centre of the circumference including the sector desired has to be moved. The implemented solution in the circular class is to divide the trigonometric circumference into eight zones, each one of 45^o and drag the centre of each sector according to its location.

In order to know the sector location the 'cumulative angle' is the reference adopted [2]. Depending on its location it is dragged to the point which is considered to be the centre. This movement must be accumulative, i.e. if two sectors are in the same area we cannot apply a fix step moving because this will result in superimposed sectors in the final figure appearing in the navigator.

This technique of cumulative movements presents the problem of knowing where the centre should be moved to. For instance, in the first zone (from 0^o to

[2] The starting angle in a sector is the sum of the length of the preceding arches

45^o)if we cumulatively move the centre along the axis X all the sectors starting at that point will be superimposed, and therefore they should be moved along the axis Y.

The applet gdAPPblp has the function of representing the scatter, line and column charts. This task is executed in conjunction with the class gdXMLblp. Despite its reduced size (less than 5KB between the applet and the class) has the following characteristics:

– It allows to change the font size. For example, the chart title can have a bigger font size than the text in the legend (this is a common characteristic to all applets)
– It adjust the scale of the axis Y. The method realises a direct interpretation of the height of the column, i.e., if the value to be represented is equal to 20 the column will be 20 pixels high. This results in some of the charts having a deviated height-width ratio. To avoid this the input values are corrected so that the length of axis Y is half of that of axis X.

3 Conclusions

One of the most important aspects of this project is the size of the applets interpreting the XML files. For this reason, much effort in elaborating the coding has been derived to their optimisation. In general terms, reducing the size of the client involves more work in the development of the server component.

Another important aspect in the client element is the fact of not using Java classes to handle XML documents. Instead we have created the classes gdXML-blp and gdXMLsec, both being able to play the roles of XML interpreters, DOM (Document Object Model) handlers and including representation methods. Also it is noticeable that the use of a DOM neither of a XML interpreter is never necessary.

With a modem connection of 56600bits/s (this is a minimal requirement nowadays) these applets can be downloaded in less than a second. To this we should add the time to open the website which contains them, and obviously, the execution time of this applet in the client. Generally the total waiting time, since the graphic is requested to when appears on screen is slightly over one second.

This work was partially funded by the C.I.C.Y.T. under the projects TEL-1999-0958 (Plan Nacional de I+D) and TEL99-0335-C04-03 (Plan Nacional de I+D).

References

1. Jan Christian Herlitz, *Drawml and svg*, Proceedings of the First XML Europe conference (XML Europe 99), 1999, Granada, Spain, 27-30 April, pp. 61–70.
2. Mark Roberts, *Graphic Element Markup*, Proceedings of the First XML Europe conference (XML Europe 99), 1999, Granada, Spain, 27-30 April, pp. 547–557.

3. Javier Rodeiro and Gabriel Pérez, *Generación de gráficos en arquitecturas cliente-servidor mediante xml*, Proceedings of XI Congreso Español de Informática Gráfica, 2001, Girona, Spain, 4-6 July.