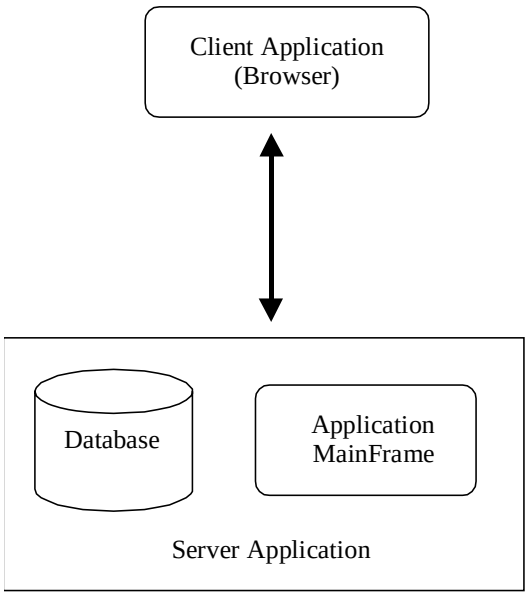
7th International Conference on DEVELOPMENT AND APPLICATION SYSTEMS
Suceava, Romania, May 27 – 29, 2004

N-TIER APPLICATIONS. XML-XSLT IN DATA TRAFFIC OPTIMISATION

Cătălin CERBULESCU[1], Monica CERBULESCU[2]
[1)] *Faculty of Automation, Computer and Electronics, Craiova*
ccerbulescu@nt.comp-craiova.ro
[2)] *Carol I College, Craiova*
mcerbulescu@hotmail.com

Abstract: *The paper presents an N-tier development model for Web applications. The purpose of the approach is minimising data processing on the server tier and moving this to the client tier. The XML data is processed thorough XSL or client scripting, using XSL or client script templates that access either the XSL document root either the XML document root. The data presentation is the task of the fat-client. The Applications and/or Web Server Level queries the database and stores the result in a XML Data Island. The XML Data Island is send to the client either through a inline XML, embedded in the HTML document either through an XML data source url, attached to a working session. The XML data, received by the client, are exploited by client scripting or by XSL files, generated, on the server, during the current session. The client (browser) switches to the fat-client situation. If the client browser does not have the capabilities of handling XML, the presentation is fully generated on the server, through server scripting and then send to the client. The client switches in the thin-client situation.*
Keywords: *XML, XSL, traffic optimisation, fat-client, thin-client, XSL root, XML root.*

2-tier Application Development Model



Client / Server application development model

Client-Server applications represents an architectural method of developing applications witch has the purpose of sending information, stored on server, to an client-user.

Client/Server (C/S) is the general term through we can refer those systems who:

1. have as a physical support a computer network and

The client initiates the contact (transaction) with the server application, normally on another machine in order to access some specific functions. The server delivers services requested by server.

Although the C/S term was frequently associated with the situation of a computer connected to a database server, it refers the logical modelling of an application, in the design phase so that it can be done a clear division of the tasks on levels, tiers or client/server.

In a typical C/S application, data presentation task is made by the client application, through a GUI while the server application ensures the database access.

On a large scale, the C/S architecture is the most common development model used in applications in the last 20 years. Today it is

obviously that it is overtake. This is due to the need

1. to process larger quantities of data but also
2. due to the frequent changes of the data processing algorithms.

Those facts require changes to the level of the Server and/or Client.

Basically, this is the so-called ”fat client” – ”thin client” dilemma. It consists in establishing the place (client or server) witch ensures the major parts of the data processing.

If the larger part of the data processing is made on server (fat server - thin client) we obtain:

- a server overloading in witch the machine respond to fewer clients in the time unit;
- the server ensures data store but also data processing;
- fewer tasks to execute for the client;
- low network data traffic (the client makes requests, he does not execute any kind of data processing);
- in the case of a Internet application, the code for the client (client scripting) has small dimension and it is easy to load on the client (browser);
- algorithms changes can be made only on the server with low costs. In the case of an Internet application algorithms changes can be easily made both on server and client, on low costs;
- client applications can be designed so that can be platform-independent.

If the larger part of the data processing is made on client (fat client - thin server) we have:

- client overloading;
- the server does, besides the data storing, minimum data processing;
- a task release for the server, this can trait a larger clients number in the time unit;
- low network data traffic (the client runs data processing and the server only serves data);
- in the case of an Internet application
- the client side scripts has a greater size and can be hard to load;
- algorithms changes are easy to made, with low costs
- in the case of an non-Internet application
    - algorithms changes are made on client side, with high costs because
    - the client applications cannot be designed so that can be platform-independent. Although, Java can help to design such kind of applications.

As shown above, the programming language is another major problem that needs to be solved when designing non-Web C/S applications. That’s because the client application, usual developed in a visual medium, must run on different platforms and being platform independent. It reaches the situation when an application, developed for a company cannot be used for another company. So, the software development became less profitable.

Some of the important elements followed in designing a C/S application are:

- the easiest the client uses the application the more complicated is the architecture of the C/S system;
- the server must be able to trait a large number of requests so that the systems performances are diminished by the network performances and not by the application;
- the use of an RMI architecture must be take in consideration in solving the “fat-client” problem. This approach will cross the standard C/S application border and became a “distributed computing” approach and 3 or n-Tier system architecture.

3-Tier Application Development Model

If an organisation needs to make important changes in the data processing system, it must change the system architecture. A well suitable architecture can lead to system easy and fast adapting to the new requirements, no matter if that means more users or new rules.

Right architecture is the key of the new client/server systems. For the most of the cases, the right choice is the N-Tier client/server architecture.

The basic 3-tier application development model is presented in the Figure 1.

C/S applications disadvantages and limitations, presented above, lead to the development of a new application architecture on 3 or more tiers.

Basically, an application on several tiers leaves the client in the situation of a “thin-client”, so

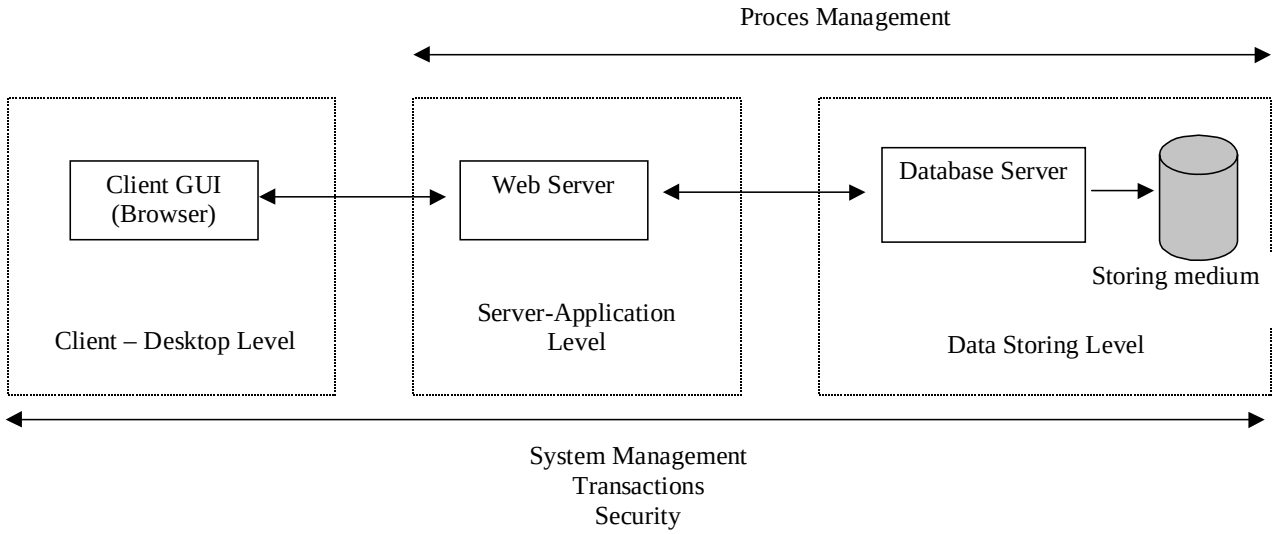


Figure 1. 3-tier application development model

that level, the most platform-dependent, is easy to implement. For this purpose can be used, for example, a browser developed for a particular platform.

Client Tier

The application client tier is responsible with data presentation, handling client events and GUI. Most of the algorithms stored here were move mostly on the next tier: the application server.

A typical implementation of this tier uses applets or client scripts. The developer can choose between one of the approaches depending on application requirements and advantages and disadvantages of each approach. The client tier, also present in the C/S model, was switched in the situation of a ”thin-client”.

Application Server Tier

The application server tier, new added in the system architecture, was not explicitly present in the C/S model. This new level is the system key. Objects that implement algorithms are stored here. The tier protects data from being accessed directly by clients. Also, it contains components that can be accessed directly by clients, on the client tier. CORBA, as an applications integration technology tend to be more used at this level.

Data Storing Level

On the data storing level it can be used various system types, from relational to non-relational databases.

The borders between tiers are logical, not physical, each one of them can run on the same machine. The only restriction is that the system to be more structured and the borders between tiers being well defined and practically represented by the composing object interfaces. The system manager, based on specific conditions for each application imposes those.

The advantages of using a 3-tier model for developing applications are:

1. separation of the 3 tiers. With this separation, more clients have access to a larger variety of server applications. Mainly advantages for client applications are: fast development by reusing algorithms components and a shorter testing phase because the server components were already tested;
2. re-defining the data storing strategy does not affect the client, this one accessing data from a well defined and well designed interface witch embeds all storing details;
3. objects manipulating data processing algorithms must be stored as closer as it can to the data storing medium. Ideal, on the same machine. This way the network traffic was reduced because this is the application zone with the most intense data traffic.

4. unlike the C/S system architecture in witch only data were accessible to the public, now the public can access services;
5. the servers, as more secure systems that can ensure a better protection and security of the data are much easy to manage and maintain. Although, as a distributed applications, data protection and access control are important parts of the system. In a simple way, the 0 level of security ensures authentication, authorisation and data cryptography.
6. related to the system upgrade, it is easy to change a soft component on the server than to deliver to the numerous clients new versions of the application.

N-Tier Application Development Model

A distributed N-Tier application is:
- a framework for providing a flexible, distributed computing environment, that can take full advantage of the infrastructure and resources you currently have, while preparing for whatever changes the future brings;
- a variation on the familiar Client / Server computing model, which uses Internet / Intranet related technology, to maximise return on investment and existing skill sets, while providing a reliable, flexible framework for change and growth;
- a method for centralising control over increasingly critical corporate information, while encouraging departmental innovation and maximising supplier and customer input.

Distribute N-Tier applications refers to the use of any combination number between hardware levels and/or software levels with the purpose to deliver a structured collection of information services.

Any number of levels can exist: client, interface, agent, transaction, data server etc. Also, that level operates as logical units, on each machine or on a various number of machines. This leads to a greater flexibility and scalability of the system.
This approach for an application permits that:
- a bigger parts of the application is eliminated from the client side, leading to the “thin client”;
- an increasing number of levels between client and data processing;
- the client level task is handling the GUI;
- integration of various resource collection in a whole system.

Distribute N-Tier applications can be developed using a large variety of programming languages, operating system and platforms.
The purpose of this approach is that it permits to each application level to be managed, installed, extended absolutely independent from others levels.
Java, as a programming language and virtual machine, is a new type of client in 2 or N-Tier systems. Because the new client can run on any computer and any platform, the software development is made not only for 2-Tier application but for N-Tier applications.

Xml Data Island

XML Data island is a XML document witch exists inside a HTML document. This concept allows data processing from the XML document. The beginning of the XML Data Island is marked with the XML tag and the ID attribute ensures a name that can be used to refer it.
Embedding XML Data Island in a HTML document can be done by:
1. using XML element in a HTML document;
2. overloading the HTML SCRIPT element.

The use of the XML element in a HTML document can be made by:
- XML data can exist inline, inside the XML tag, as below

```
<XML ID="XMLID">
  <XMLDATA>
    <DATA>TEXT</DATA>
  </XMLDATA>
</XML>
```

- XML element can have a SRC attribut, witch value is the XML source data url, as below<XML SRC="http://file_url/xml_name.xml"></XML>

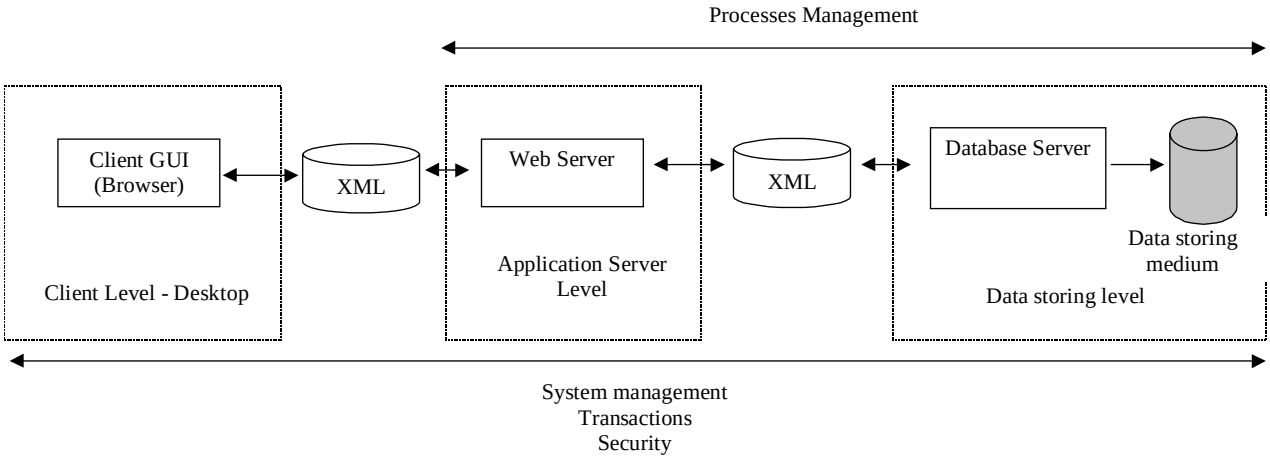


Figure 2. 3-Tier XML based architecture for web applications

XML element is present in the HTML DOM, in the *all* collection and is being viewed by browsers as a tree node. XML dates from inside the XML element can be accessed through the XMLDocument property of the XML element.
XMLDocument property returns the XML tree root node, from the XML element or by the XML referred through the SRC attribute value. From this root, it can be browse through XML Data Island using XML DOM.
By overloading the HTML SCRIPT element, the XML dates can be accessed by calling the XMLDocument property for the SCRIPT object.
The XML data can be generated from an existing database, using an N-Tier application model. With XML, structured data can be kept separated from the presentation and display level.

XSL

An XSL transformation is usually used for browsing XML documents and leads to the generation of a new XML tree. This clear separation between the document source and the resulted data trees accomplish the XML purpose, which is separate the content from presentation. So, it allows both the XML grammar and XML source structure to be independent from both the language and presentation structure.
The XSL document contains a template with the desired resulting structure and identifies dates from the XML source witch can be inserted in this template.
The XSL ensures data processing from real documents, recursive data or with a high irregularity. Templates fragments are defined and the XSL processor combines the result of this fragments into a finally tree based on the XML data. Each template fragment declares the type and the source node context to witch it is assigned, allowing the XSL processor to establish a relation between the source nodes and the templates fragments.
XSL transformation responds to the XML requirements:

- allows XML data display, transforming the XML into a displaying suitable structure;
- allows the direct browse of the XML files through high-level browsers facilities;
- XSL transformations can be executed on the server so that the HTML documents containing XML data can be served to the low-level browsers;
- schema transformation. The transforming process is independent from any particular grammar and can be used to translate XML data from one schema to another;
- XML conversion through query, sorting and filtering sentences.

Xml/Xsl. Solution For Data Traffic Optimisation In 3-Tier Applications

XML offers a robust solution for 3-Tier applications, as it can be observed from the Figure 2.

The data, stored in various forms are converted in XML format, tacked and processed by the Web server and sent to the client in the form of a ”data island”, inline, as XML documents inside HTML documents or as a XML source url.
The advantage of this solution became obviously when the client uses a browser with the capability of handling XML, through XSL or client scripts that access the XML or XSL document root. As it can be observed by the Figure 2, at the level border, only XML format dates are travelling.
By taking in consideration the frequent situation in witch servers must trait a huge number of client requests, an increasing data traffic takes place between the levels Data Storing and Application-Server but also between Application-Server and Client.
The data traffic optimisation inside the application brings in discussion:
1. Loading from the database server of an optimal data quantity and sending data to the Application-Server level. The optimisation principle can be refer here:

- a single data loading from the databases, of a maximal data quantity witch can be used in data processing either
- on a several loads of small quantities of data, the requests to the database server being made by the Application level.

Depending on the system characteristics and on the application liberty degrees, it can be choose one of the two above described techniques;
2. optimising the communication between the Client level and the Application level. This can be achieved:

- through a single request from the client and a large quantity data response from the Application-Server level;
- through a series of client requests followed by small quantity data response.

In order to optimise the system optimisation it can be consider a constant product between the number of server-client-established liaisons and the data quantity send by such a link.
Web applications that needs to handle a large number of client requests, it is desired to minimise the data processing made on server and moving those to the client.
The purpose of the approach is minimising data processing on the server tier and moving this to the client tier. This is possible because XML became a standard and more and more browsers have the capability of processing XML through client scripting.
The server response to the client will be in HTML format and it will contain or not embedded XML data. Using data island, the client became fat-client, with the advantages and disadvantages exposed above. The minimum number of the responses of a the server through the client will be:
1, for the situation of using inline data island. The response contains XML as embedded data in HTML;
2, for the situation in witch data island are presented as an XML source url. The response is only in HTML format and the XML dates are loaded separately.
During a working session, uniquely attached to a client, the server queries the database (the dates are stored in no matter what form) and the result will be stored in a temporary XML file and eventually an XSL file. The files are attached to the session and can have the lifetime of the session that creates them.
We consider a frequent situation in witch a client needs to access a particular set of data from server. It is also very probable that the client will require the control over the selected data manipulation.
As an example, we suppose a client that accesses a university database and selects from it a set of students. The required students will be extracted from the database (stored in any format !) and stored either in a XML data-island either inline, in the HTML file.
In order to minimise the further number of server access, the client must request a maximum number of information about the students: name, birth date, registers data, etc. Much more, the server must generate the XSL or scripts that can cover the whole types of operations that the client may need. For the existing database, the application can have, for example, templates for the following operations: selecting students based on one criteria, selecting result on several courses etc. Once the

data arrived at the client, this can select the desired presentation mode: either the display of whole data or, more probable, selecting from the set of some students, some particular data, such as data register regarding general averages marks, general averages marks sorted by courses, etc.
The combination of XSL-client scripts that manipulates XML data through DOM is enough to cover the whole area of operations that the client might need.
For general presentations, for example: the whole data selected from the database, an XSL can be used. The server generates this XSL at the moment he generates the XML. The XSL is attached to the working session and has, eventually, his lifetime.
For specific operations, like displaying data for students with average mark greater than a desired mark, it is necessary to manipulate the XML through client scripting. Client scripts access either the XSL document root, either the XML document root, as a consequence, using or not XSL transformations.
So the server manipulates a part of the information stored in the database, analyses the browser capabilities and sends data to the client. Depending of the browser capabilities of handling XML through scripting and/or XSL, we have 2 possible situations that server need to handle in order to obtain, at the client, the desired presentation:
1. the browser has the capabilities of processing server sended XML data and obtain the desired result. The browser manipulates the XML data through client scriting that access the XML document root or XSL document root. By the C/S traffic point of view this is the “fat-client” situation. The task of the client task is to process and present them. The task of the server is:

- querying the database;
- converting the query result in XML format;
- choosing the way of sending the XML data to the client, through data-island or inline XML file;
- sending XML data to the client;
- generating the XSL files, for transforming XML;
- generating HTML response page to the client and the client scripts that will manipulate XML data;

This is the situation that allows data traffic optimisation between client and server. The client has all the data he needs to present but also has all the scripts to manipulate them, without the server intervention.
Depending of the databases available for the application, that can pre-generate a series XSL or client scripts templates that will be inserted in HTML response to the client. The operations implemented by the templates will cover a large area of client presentation requests. It will achieved the situation when, based on the client request, the server sends to this the XML data but also the templates for the desired operations.
2. the browser does not have the capability of presenting XML data through client scripts that access the XML or XSL document root. Also, the browser does not have the possibility of loading an XSL from the server. This case, the server makes all the data manipulations, through server scripting or XSL templates. By the point of view of the C/S data traffic it is a “thin-client” situation. The server has the task of processing and sending data and the client has the task of displaying data. Any new client request of presenting data is sended to the server, analysed by this and the response is sended to the client. In this case, the use of XML does not have major advantages.

Conclusions

The proposed model (Figure 2), combining the 3-Tier applications and XML/XSL advantages, can be successfully used in Web applications, in witch the client uses an browser that have the capabilities of handling XML data through client scripting. The script accesses the XML or XSL document root. The client switches to the “fat-client” situation, data presentation being exclusively his task. The server has only the task of selecting from the database an maximum amount of data and to send them to the client using one of the two XML Data Island formats: inline XML Data Island or a XML data source url.

The server also generates an amount of XSL or client script templates that will be used to manipulate the XML data. The templates will be generated according to the types of data manipulation algorithms that the client will need in order to present data.

As a result of this approach, the Application-Server and Data Storing levels are free from the task of processing data, with the following advantages:

- a low server response number to a client, in the situation in witch a client receives all the dates needed in a presentation. The response will include, besides the HTML presenting data format, the XML data and XSL;
- on a client request, the server selects from the database a maximal set of information, sends them to the client, after what the communication between those two ceases;
- in the case o using the model in a Intranet, having the same client browser type, the Application-Server level don't need to detect the browser type that makes the request, in order to analyse his capabilities. So, the application response time decreases.

If the browser does not have the capabilities of handling XML, the client became a "thin-client" and the use of XML does not have major advantages.

The solution has major advantages in Intranet Applications, where all the clients use the same browser and the maximum data manipulation algorithms can be know a-priori by the server.

Reference

[1] Jane Sturm, (2000), *Developing XML solutions*, Microsoft Press

[2] MSDN Library (2003), *N-Tier Development Model*

[3] MSDN Library, (2003), *Creating a Three-Tier Web Site*

[4] William Pardi, (1999), *XML in Action. Web Technology*. Microsoft Press

[5] www.n-tier.com, *The N-Tier Re*volution