BUILDING WEB APPLICATIONS WITH XQUERY
Integrating technologies in web development

Javier J. Gutiérrez, María J. Escalona, Manuel Mejías, Jesús Torres
Department of Computer Languages and Systems.University of Seville.

Keywords: XQuery, web development, web engineering, portal strategies, XML, Data management, technologies integration.

Abstract: Today, it is needed to apply a set of heterogeneous technologies to implement every layer or element in a web application. These technologies must be combined and must work together. This one implies the need for heterogeneous development teams with heterogeneous formation and high costs in tools and formation. This work shows how XML with XQuery could be a valid technology to unify the technologies used in web development. This works shows how XML and XQuery is a valid selection to unify the used technologies in web development. Thus, it is possible decrease costs in tools and formation applying only one technology in web development. To justify why XML with XQuery is a valid technology to implement a whole system this work shows, at first time, the main characteristics of XQuery focused in web development. At second time, this work shows how to apply those characteristics in a web development and how to implement every layer or component of a web application with XQuery. Finally, this work exposes a brief overview about the open-source tools available to implement a web application with XQuery.

1 INTRODUCTION

This section exposes the problem described in this work and solution proposed. Next sections describe how to apply this solution, its advantages and inconveniences, open-source tools to apply it and conclusions and future work.

1.1 The land of 1,000 technologies

The song "The land of 1,000 dances" enumerates a long list of modern dances in the 60s. Now, in 21th century, we still have 1,000 dances and maybe more, and we also have 1,000 technologies and maybe more, to build a web application.

Web software systems have become more important and complex software. Web software is based on novel computing technologies. Sophistication of web systems has increased rapidly. First web systems were static set of information stored into static HTML pages. Second generation were simple applications that accepted data from HTML pages and process or stored these data. Today, web systems are sophisticated, interactive programs with complex user interfaces and large amount of integrated back-end software (Offutt, 2002), (Ye-Wu, 2004).

This fact brings the need for using a heterogeneous set of technologies and tools to develop a web system. Web software is build with many different technologies like: scripting languages that run on the client, interpretive languages that run on the server, general purpose programming languages, data manipulation languages and databases (Ye-Wu, 2002). Table 1 shows an example of these technologies.

Table 1: An overview of technologies used in web development.

Clients	MS Internet Explorer, Opera, Firebird, Amaya, Safari, etc.
User Interface	HTML, XHTML, DHTML, CSS, Macromedia Flash, XML-XSLT, etc.
User components	JavaScript, VBScript, ActiveX, Java Applets, etc.
Server components	Java Servlets, PHP, ASP, ASP.NET, Perl, Webservices, etc.
Data components	Access, SQL Server, Oracle, MySQL, PostgreSQL, etc.

This heterogeneous set of technologies (and many more) have to cooperate together to implement a

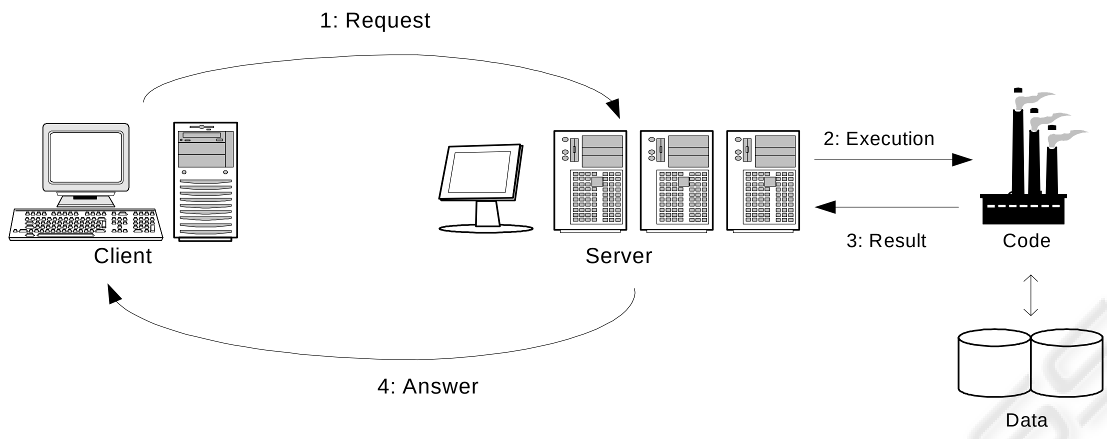


Figure 1: Web application layers

web application. So, a web application results in a multi-platform software environment. This fact brings many problems:

- Many experts in several technologies are needed.
- Bigger cost in personal and formation.
- Heterogeneous technologies make hard testing process (Offutt, 2002).

These problems can be avoided unifying the technologies of web development. Next section introduces the pair XML and XQuery as an integrated tool to built web applications.

1.2 An integration solution

Classic layers of a web application are: user interface, business logic and data management. Figure 1 shows these layers and the communications among them.

Many of technologies are focused in one of the layers in figure 1, as showed in table 1.

A first integrator effort has been made with XML (World, 2004). Although originally designed for large-scale electronic publishing, XML plays an increasingly important role in data exchange on the Web. Some authors expect that XML will become the lingua franca of the Web, eventually replacing HTML (Mignet, 2003). In this way, there is a proposal to apply web syntactical rules over HTML called XHTML (XHTML, 2004).

XML is good to store information and to describe the semantic of that information. In web systems, XML can be used to describe the information, business rules, user interfaces, configuration of the application, etc. Although data is stored in a relational database, it can be extracted in an XML structure (Marchal, 2001).

However, XML cannot describe the active parts of a web application. XML resolves the half of the problem only. We still need technologies to manipulate and process XML. Another problem is that, nowadays there is also a heterogeneous set of technologies to manipulate XML. Table 2 shows and example of open-source tools for Java technology (McLaughlin, 2002).

Table 2: Techniques and tools to XML process

Techniques	**Tools**
APIs	DOM, SAX
XML-Binding	JAXB, XML-Beans
Dynamic Beans	Apache Betwixt, Codehaus XStream
Others	Apache Digester, Apache XMLIO

Each one of these technologies is oriented to a concrete scenario. For example, to create objects from XML when XML-Schemas or DTDs are available, XML-Binding tools use to be the best option. But, if the task is to load a simple configuration file described with XML tags, lighter tools are a better options. Light tools are Apache XML-IO or SAX API.

A modern technology that might unify the process of XML in the different layers of a web application is XQuery (W3C1, 2002). Next section introduces XQuery and shows how to built every layer of a web system.

2 AN INTRODUCTION TO XQUERY

XQuery is a powerful query language for XML data. Some of its strong points are: it has native support for handling over forty built-in data types, powerful constructs for bulk data processing, like expressing joins, aggregation, and so on, support for text manipulation, and a notion of document ordering that provides a foundation for a variety of interesting document-oriented queries (Brundage, 2004).

After several years of development in the W3C, XQuery is starting to gain significant traction as a language for querying and transforming XML data. Though the W3C XQuery (W3C1, 2002) specification has not yet attained recommendation status, and the definition of the language has not entirely stabilized, it is already beginning to appear in a variety of commercial and open-source products. Examples include XML database systems, XML document repositories, and XML-based data integration offerings.

An XQuery expression is composed of a maximum of 5 different sections, each one with its own mission. All sections are showed and described in figure 2.

An XQuery program or script is a just an expression with some optional function and other definitions. So 3+4 is a complete, valid XQuery program that is evaluated to the integer 7.

Next section describes the main characteristics of XQuery language that might be applied to web systems development.

2.1 Main characteristics to web development

This section does not want to be a complete description of XQuery. Nowadays there are many good books to learn all characteristics of XQuery and how to apply them, like (Brundage, 2004), (Katz, 2003) and the documents of the standard in (W3C1, 2002).

The main goal of this section is to expose a resume of the characteristics presented in XQuery that makes itself adequate to web development. Characteristics studied in this section are listed in table 3.

Table 3: XQuery characteristics to web development.

– Queries in XML. – XML transformation in HTML or XML. – Functions. – External functions. – Open-source tools available.

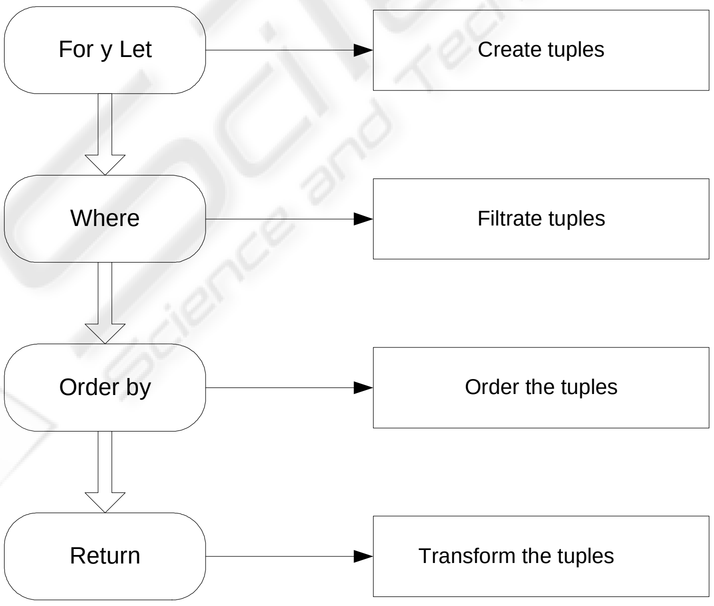
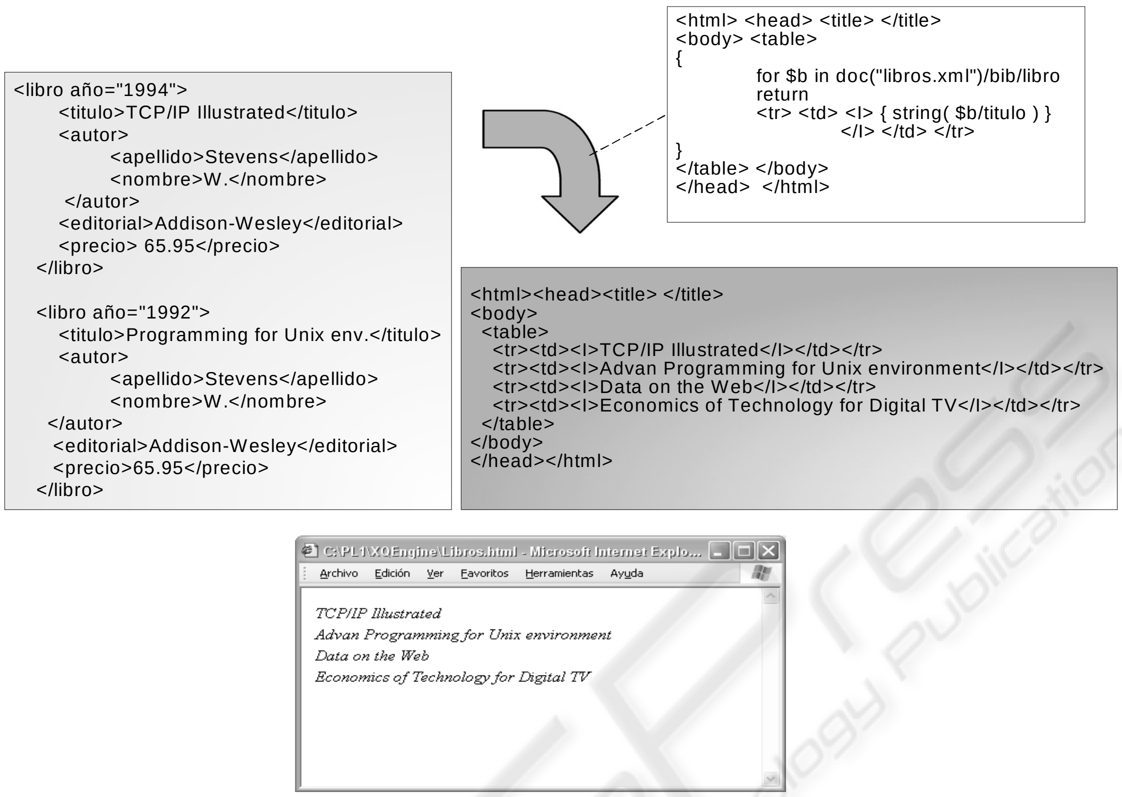


Figure 2: Sections in a XQuery expression



```
<libro año="1994">
      <titulo>TCP/IP Illustrated</titulo>
      <autor>
              <apellido>Stevens</apellido>
              <nombre>W.</nombre>
       </autor>
      <editorial>Addison-Wesley</editorial>
      <precio> 65.95</precio>
   </libro>

   <libro año="1992">
      <titulo>Programming for Unix env.</titulo>
      <autor>
              <apellido>Stevens</apellido>
              <nombre>W.</nombre>
     </autor>
     <editorial>Addison-Wesley</editorial>
     <precio>65.95</precio>
   </libro>
```

```
<html> <head> <title> </title>
<body> <table>
{
            for $b in doc("libros.xml")/bib/libro
            return
            <tr> <td> <I> { string( $b/titulo ) }
                        </I> </td> </tr>
}
</table> </body>
</head>  </html>
```

```
<html><head><title> </title>
<body>
  <table>
    <tr><td><I>TCP/IP Illustrated</I></td></tr>
    <tr><td><I>Advan Programming for Unix environment</I></td></tr>
    <tr><td><I>Data on the Web</I></td></tr>
    <tr><td><I>Economics of Technology for Digital TV</I></td></tr>
  </table>
</body>
</head></html>
```

C:\PL1\XQEngine\Libros.html - Microsoft Internet Explo...
Archivo Edición Ver Favoritos Herramientas Ayuda

TCP/IP Illustrated
Advan Programming for Unix environment
Data on the Web
Economics of Technology for Digital TV

Figure 3: Transform an XML structure into HTML.

Characteristics in table 3 are described in sections below.

2.2 XML search

XQuery is a concise but flexible query language for XML. So, its main goal is to specify queries and to perform it over a XML dataset. XQuery is based on the structure of XML and leverages this structure to provide query capabilities for the same range of data that XML stores.

It is possible to include many search parameters in a query. It is possible to search for tags without matter their location into XML structure, or to search for tags inside a concrete XML structure. It is also possible to obtain new XML structures from exiting XML structures. For example, from a XML structure containing books with the editorial of each book, it is possible to generate a new XML structure with editorials, and all books edited by each editorial.

2.3 XML transformation

XSLT allows to describe XML transformations, that is, operations that take an XML structure and generates one or several XML structures as output (Robert, 2001). Some authors estimate that probably 80 percent of the actual usage of XSLT is for transforming XML to HTML. This might be handled by treating the result document as a well-formed XML tree, with the transformation being followed by a serialization phase that translates this tree into an HTML or XHTML output file (Katz, 2003).

Figure 3 shows an XQuery expression that transform a XML structure into an HTML structure. Figure 3 also shows how is rendered that HTML structure into a web browser.

One of the main problems until now is that HTML does not satisfied XML syntactical rules. This problem makes hard to transform an XML structure in HTML, and does not allow using all possibilities offered by HTML. As seen before, there is a new standard called XHTML (XHTML, 2004) that satisfied XML syntactical rules. This standard is been rapidly implemented with main browsers and, nowadays it is possible to use it in web development.

XQuery is more easy and intuitive than XSLT. Transforms described with XQuery expressions are easier to understand, maintenance and modify than transform expressions written in XSLT.

2.4 Functions

When an XQuery expression becomes large and complex, it is often much easier to understand if it is divided into functions. These functions can be reused in other parts of the expression. Functions can be recursive, which is a good help for processing the recursive structure of XML (Brundage, 2004).

It is possible to organize functions in library modules. These modules can be used and imported by any query. Every module in XQuery is either a main module, which contains a query body to be evaluated, or a library module, which has a module declaration but no query body. A library module begins with a module declaration, which provides a URI that identifies the module for imports.

Functions defined in library modules are namespace-qualified. This means that functions with same name can exist into different namespaces. Any module can import another module using a module import, which specifies the URI of the module to be imported. It may also specify the location where the module can be found.

2.5 External functions

XQuery allows calls to external functions implemented in the same language than the environment under XQuery engine is in execution, such as Java or C#. The environment can provide external functions and variables to XQuery. To access to external elements, a query must declare them in its prologue. The mechanism by which this is done varies from one implementation to the next. XQuery itself provides only the syntax to use when declaring these externals, shown in table 4.

Table 4: External function definition.

```
define function out($v as xs:integer)
as xs:integer external
```

External functions are equal to ordinary user-defined functions except that the external keyword is used instead of a function body. XQuery does not specify how such functions and variables are made available by the external environment, or how function parameters and arguments are converted between the external environment and XQuery, so it depends of the vendor implementation.

3 APPLYING XQUERY IN WEB DEVELOPMENT

XML is one of the most important technologies in web development. One of the main objectives in design of XQuery is to allow that XQuery can be applied like a XML manipulation and transformation language. Thus XQuery offers tools to work with XML. Moreover, XQuery can be successfully applied in environments where XML works. Some of the most important applications in web environments are showed in table 5.

Table 5: XQuery in web development

- Management XML native databases in the same way than SQL manages relational databases. - Generate web interfaces in the same way than XSLT. - Call web-services. - Syndication with RSS.

In section 2, main characteristic of XQuery to web development have been defined. This section shows how to apply those characteristics to implement each layer showed in figure 1.

3.1 User interface

This work has described, in sections before, how it is possible to build web user interfaces using XQuery in a similar way than XSLT. This generation interface technique has been improvement with the inclusion of XHTML and CSS in web browsers.

The main differences between XSLT and XQuery are of two kinds. First, they have different requirements, and therefore a design decision that was appropriate for XSLT would not necessarily be right for XQuery, and vice versa. The second kind of difference results from their being designed by different people from different communities and computing traditions, with different beliefs about what constitutes good design, and different experiences as to what works well and what doesn't.
There is one important difference between XQuery and most template systems: With XQuery, it is also possible to define functions that return HTML fragments, and to pass those fragments though to other functions. With most template systems, it is

only possible to create output fragments as strings, and pass them around as strings.

3.2 Business logic

Functions and external functions are the main tools to build business logic with XQuery. Although XQuery main objective is to process XML, it also has a complete type system

The type system of XQuery is the same than schema type system (Robert, 2001). There are two sets of types in XQuery. First are the built-in types that are available in any query or function. Second are the types imported into a query from a specific XML-Schema or DTD.

Not all data in XQuery must be XML structures. It is possible to write expressions that return an <xs:integer> or define parameters in a function like <xs:char>. XQuery can also works with untyped data, strongly typed data, or mixtures of the two. When XQuery has not information to know the type of the data is processing, XQuery applies a set of rules to infer an appropriate type

This one allows writing business logic in a similar way that, for example, PHP, without object orientation.

3.3 Data layer

To implement data layer, it is possible to use a database server which supports recovering of data in XML structures. However use of SQL statements will be mandatory. Nowadays XQuery has no direct integration with SQL and, furthermore, the objective is to minimize the number of different technologies applied. By this reason, we suggested to implement data layer with native XML databases.

The concept of native XML database is not a precise technical term. Essentially, this concept refers to a database designed specifically for the storage and retrieval of XML documents, and the term is used to contrast such a system with a database that merely provides an XML interface to data whose intrinsic data model is something different (Katz, 2003).

Main characteristic of a native XML database is the use of the XML data model at the query language interface to the system, and in all its other interfaces. We have reviewed some of these interfaces and seen how XQuery represents only a small part of the facilities that a real system needs to offer.

Native XML databases, in the same way than classic relational databases, cannot be applied to all kind of systems. Native XML databases are designed for use in systems when information can be expressed with XML structures in an easy and natural way.

An example of a native XML database with XQuery support can be found in (Exist, 2003)

3.5 Web architecture based in XQuery

Architecture of a web system developed with XML and XQuery is showed in figure 4 and detailed in next paragraphs.

Data is stored into several XML structures. These structures are stored and indexed in a native XML database or in a collection of XML files.

A set of XQuery expressions selects data needed for business logic and builds new XML structures with selected data.

Then, selected data is sent to business layer. In this layer, a set of XQuery functions or external functions is applied over these XML structures to add new elements, for example totals or special offers. A final XML structure contained all information to display to the user is built and sent to user interface layer.

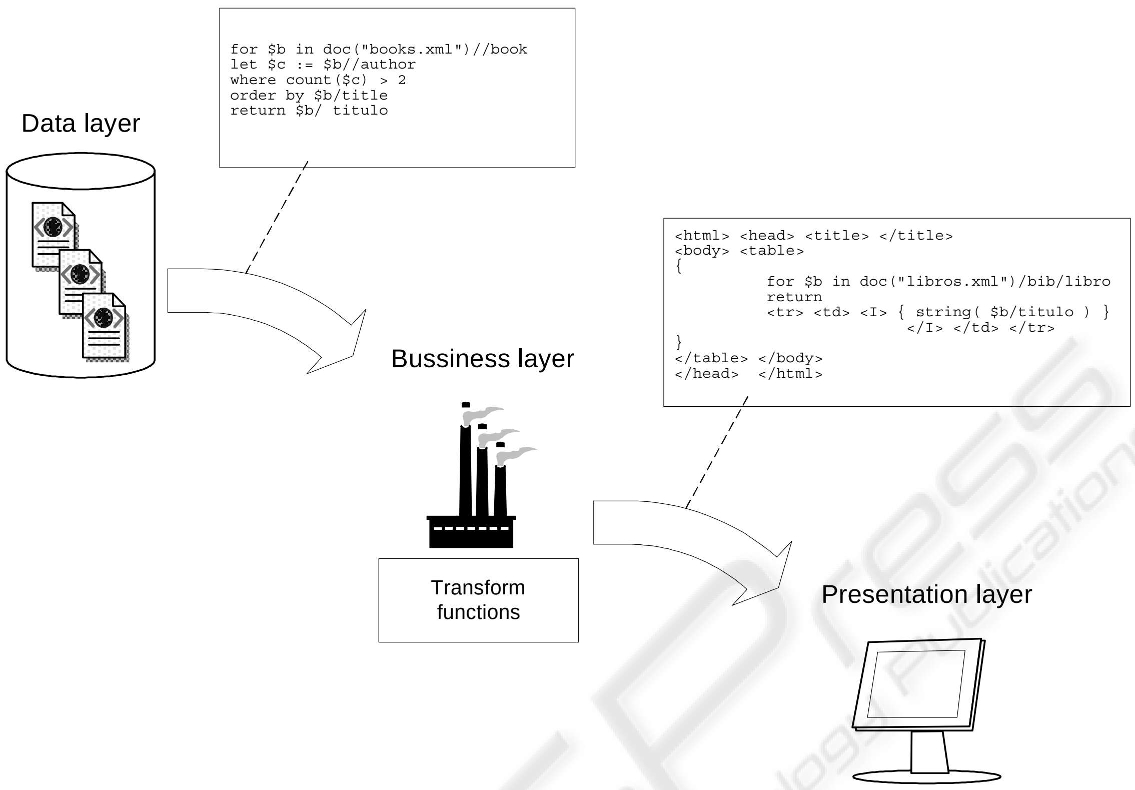
Finally other XQuery expressions are applied over final structure to transform it into a browser-compatible structure, for example HTML or XHTML, and it is sent to the client to be displayed.

Table 6 describes an application of this architecture implemented in a book on-line catalogue similar to Amazon.

Table 6: A brief description of an on-line catalogue in XQuery.

First, an XQuery expression selected XML structures of books that will be showed in the browser. Later, this structures are processed to a set of functions to add special offers (like 2x1), special prices of to add similar books. A final XML structure is generated with this information. Another XQuery expression transforms the final XML structure into HTML or XHTML.1

Next section describes open-source tools available to develop web applications with XQuery.



```
for $b in doc("books.xml")//book
let $c := $b//author
where count($c) > 2
order by $b/title
return $b/ titulo
```

```
<html> <head> <title> </title>
<body> <table>
{
          for $b in doc("libros.xml")/bib/libro
          return
          <tr> <td> <I> { string( $b/titulo ) }
                         </I> </td> </tr>
}
</table> </body>
</head>  </html>
```

Figure 4: Web architecture with XQuery.

4 A BRIEF COMPARATIVE OF OPEN-SOURCE TOOLS

Open-source community has developed very important tools in web engineering. Some examples are web servers like Apache or Tomcat (Tomcat, 2003), or languages like PHP or PERL. One of the factors of rapid and wide diffusion of open-source tools is the available of high quality tools that have became into a standard in web engineering. Thus, this section shows a brief overview of XQuery web tools

First fact to notice is the number of open-source XQuery tools is limited. Open-source XQuery engines found are (Qexo, 2003), (Qizx, 2003), (Saxon, 2003) and (XQEngine, 2003). It is out of the scope of this work to perform a complete analysis and comparative among them. This section analyzes the characteristics of each tool focused on web development.

A first analysis reveals that XQEngine and Saxon do not allow any web integration. Thus both are discarded.

Next section studies web integration capabilities of Qizx and Qexo. Both engines can be added as a module into a web server. Both generate web pages when requested by a browser. However, the process to generate output is different.

4.1 Compiling XQuery to Servlets

Qexo includes a tool to compile an XQuery expression into a Java Servlet. This Servlet can be executed in any server that supports Servlets, like Tomcat (Tomcat, 2003)

Qexo allows performing the compilation dynamically or by anticipation. Dynamic compilation is performed when an XQuery expression is compiled only when server receives the first request for that expression. Before answer, Qexo compiles expression to a Servlet if that Servlet does not exit. Then, Servlet is executed and its output is sent to client.

Anticipate compilation means that all XQuery expressions are compiled to Servlets. Then those Servlets are uploaded to the server.

4.2 Inserting XQuery into HTML pages

Qizx offers a different solution from Qexo. It is possible to include XQuery expression directly into HTML files. Qizx also offers a proprietary extension to manipulate elements from the web execution environment. This extension allows to read HTTP headers and to access to GET and POST values.

Web server has to be configured to mapping HTML files with XQuery included. When a request for a file of this type is received, server must to execute Qizx with requested file. Qizx process XQuery expressions and includes their result into HTML code.

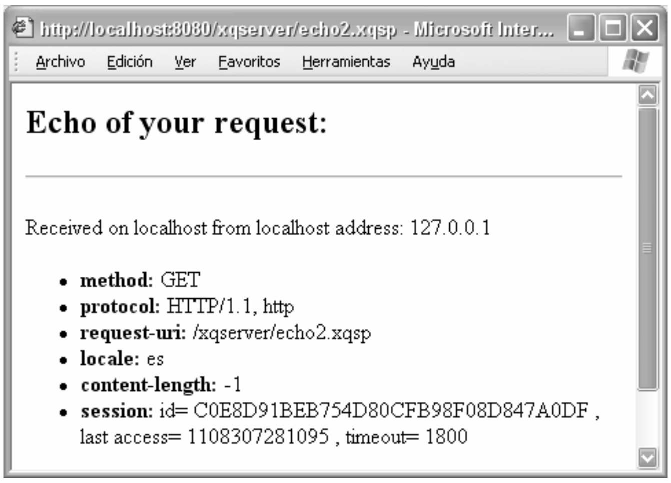
Table 7 shows an example of a HTML file with XQuery expressions. Figure 5 shows how this file is rendered into a browser.

Table 7: Example of HTML file with XQuery expressions.

```
(::pragma qizx:serialization
      media-type=text indent=no include-content-type=yes method=XML ::)

declare namespace session = "java:javax.Servlet.http.HttpSession"

<html> <body>
<h2>Echo of your request:</h2>
<hr/>
<p>Received on { request:get-server-name() } from {
    request:get-remote-host(), " address: ", request:get-remote-addr()
}</p>
<ul>
<li><b>method: </b>{ request:getMethod() }</li>
<li><b>protocol: </b>{ request:getProtocol() }, { request:getScheme() }</li>
<li><b>request-uri: </b>{ request:getRequestURI() }</li>
<li><b>locale: </b>{ request:getLocale() }</li>
<li><b>content-length: </b>{ request:getContentLength() }</li>
<li><b>session: </b>{
  let $s := request:getSession() return
  (  "id=", session:getId($s),
     ", last access=", session:getLastAccessedTime($s),
     ", timeout=", session:getMaxInactiveInterval($s),
     session:setMaxInactiveInterval($s, xs:int(10) ) )
 }
</li>
</ul>
 <hr/>
<a href="source.xqsp?src=echo">source code</a>
</body> </html>
```


http://localhost:8080/xqserver/echo2.xqsp - Microsoft Inter...
Archivo Edición Ver Favoritos Herramientas Ayuda

Echo of your request:

Received on localhost from localhost address: 127.0.0.1

- **method:** GET
- **protocol:** HTTP/1.1, http
- **request-uri:** /xqserver/echo2.xqsp
- **locale:** es
- **content-length:** -1
- **session:** id= C0E8D91BEB754D80CFB98F08D847A0DF , last access= 1108307281095 , timeout= 1800

Figure 5: HTML file with XQuery expressions.

The possibility to merge HTML with XQuery makes development of web applications easy and intuitive. However with this strategy it is possible to acquire bad habits presents in other languages like PHP. These bad habits drive to spaghetti scripting, where all layers are merged in a few big and monolithic scripts. This makes the development process hard and the maintenance process a titanic task.

Adequate strategy, in our opinion, is neither inserts XQuery into HTML, nor inserts HTML into XQuery. An application must be completely generated with XQuery and, final HTML, must be generated with XQuery expressions.

5 CONCLUTIONS

XQuery has several characteristics that allow to apply it in web development. This work has described these characteristics, how to apply them to web development, and existing open-source tools to realize them.

Thus, we conclude that today XQuery and its open-source tools can be used to build a real web system.

Two main advantages using XQuery in web development are: in first place, it is a W3C standard with mature open source tools. This open sources tools, as showed in section 4, are adaptable to web environments. In second place, XQuery is a technology that might replace a set of heterogeneous technologies. This means less human, formation and development tools costs and resources.

XQuery standard is not closed yet. It is a draft since 2002. However it is a stable draft and big changes are not expected. There are an important number of tools based in XQuery, not only open-source, but commercials too.

However XQuery draft does not included all elements needed to web development. First lack is absence of mechanism to manipulate web elements, like URL or forms. This one obligates tools to implement proprietary solutions incompatibles with other tools. There is also not a standard to define calls to external functions. Different tools are incompatible among them again.

Is XPath a valid alternative?. We do not think so. XPath (W3C, 1999) is a language for addressing parts of an XML document. XPath does not include the same flexibility than XQuery. For example, XPath returns what it found, not allowing to transform results. XPath does not support functions. Thus, XPath is one of the pillars of XQuery, but XPath, by itself, is not enough to build web applications.

We suggest a set of actions to improve XQuery as web development tool. First action will be the definition of a communication mechanism between XQuery expressions and web environment. This mechanism has to be proposed like a standard o like a section of the actual standard. Second action will be the development of new tools and the adaptation of exiting tools to this standard.

Actually, the market of web development has grown up very fast in a short time. Today there are several technologies and tools very matures. Frameworks like Struts have become in a standard and development tools like Rails over Ruby are changing the way to built web application. Thus, we suggest a third additional action. XQuery has to develop and offer to web community frameworks similar to exiting. These frameworks will allow, first, to facility the transition process to XQuery and, second, to avoid implementation the same basic concepts once and another.

REFERENCES

Brundage, Michael. 2004. *XQuery: The XML Query Language*. Addison Wesley.

Exist. 2003. http://exist.sourceforge.net/

Katz, Howard; Chamberlin, Don, et-al. 2003. *XQuery from the Experts: A Guide to the W3C XML Query Language*. Addison Wesley

Marchal, Benoît. 2001. *XML by Example*. QUE.

McLaughlin, Brett. 2002. *Java and XML Data Binding*. O'Reilly.

Mignet L, et-al. 2003. The XML Web: a First Study. *Proc. of International World Wide Web Conference*. WWW2003. Budapest. Hungary.

Offutt, Jeff. 2002. Web Software Applications Quality Attributes. *Quality Engineering in Software Technology (CONQUEST 2002)*, pages 187-198, Nuremberg, Germany.

Qexo. 2003. http://www.gnu.org/software/qexo/

Qizx/open. 2003. http://www.xfra.net/qizxopen/

Robert Gardner, John; Rendon, Zarella L. *XSLT and XPATH: A Guide to XML Transformations*. 2001. Prentice Hall.

Saxon. 2003. http://saxon.sourceforge.net/

Apache Tomcat 4.1. 2003. http://jakarta.apache.org/tomcat/

W3C. 2004. The sixth public Working Draft of XHTML 2.0. http://www.w3c.org/TR/2004/WD-xhtml2-20040722/

XQEngine. 2003. http://xqengine.sourceforge.net/

W3C XML Path Language (XPath) Version 1.0. 1999. http://www.w3.org/TR/xpath

W3C XQuery Drafts. 2002. http://www.w3.org/TR/XQuery/

W3C. 2002 XML Query Use Cases. http://www.w3.org/XML/Query, August 2002.

World Wide Web Consortium. 2004. eXtensible Markup Language (XML) 1.0. http://www.w3.org/XML/.

Ye Wu and Jeff Offutt. 2002. Modeling and Testing Web-based Applications. *ISE Technical ISE-TR-02-08.*

Ye Wu, Jeff Offutt, Xiaochen Du. 2004. Modeling and Testing of Dynamic Aspects of Web Applicationsy. *Submitted for journal publication.*