Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Efficient Web-based Application Development Tools on XML-enabled Databases

A thesis presented in partial fulfilment of the requirements for the degree

of

Master of Information Sciences

Yi Chen

Massey University

Palmerston North, New Zealand

July 2008

Table of Content

1 Introduction

1.1 Web-based Application Development

1.1.1 Web-based Applications

With the emergence of the World Wide Web also web-based applications became increasingly popular. Web-based applications bring together traditional software applications and traditional web sites. *Software applications* are computer programs that help users to perform specific tasks. They are often contrasted to system software that help to exploit and maintain the various capabilities of computers, and to programming software to assist a programmer in developing computer programs. Examples of software applications range from personal software (e.g., word processors, spreadsheets and media editors) to enterprise software (e.g., supply chain management systems, customer relationship management systems and executive information systems).

Today, most software applications in enterprises are distributed systems: the software is spread over various computers. This is most natural for enterprise software that is accessed by many users across a variety of locations. But there is also an ongoing trend to design and implement personal software as distributed systems with personal web information systems, social and collaborative software, and software as a service being recent examples. Most distributed applications are client-server applications, that is, they are built using a *client-server architecture* that distinguishes client and server software and describes how the distributed programs communicate. This client-server model enables the joint usage of resources (e.g., databases, printers) through a network. For software applications, a major part of the software can be kept in dedicated *application servers* while small client programs are installed on client computers and function as interfaces to the users. This approach has obvious advantages when it comes to maintaining and upgrading software applications. Over time there evolved the desire to standardize these client programs, e.g. by using *web browsers* that are commonly available on most client computers and frequently used to access all kinds of web sites.

Traditional *web sites* provide static content through a network such as the Internet or an intranet. Web sites are also based on a client-server model. The content is kept in web pages (mainly HTML files). Users can view the content through client programs called *web browsers.* For that the web browser sends a request to the *web server* that responds by delivering the requested web page. Communication between web browsers and web servers is based on the hypertext transfer protocol (HTTP). Examples of web browsers are Firefox, Internet Explorer, Netscape, Opera and Safari, to mention the most popular ones. Over time there emerged a desire for more involved interaction with the users. That is, web sites should be enhanced to provide dynamic content.

Both the aim to make distributed software applications web-based and the aim to make web sites interactive resulted in the development of *web-based applications*: software applications that can be accessed through a web browser. More precisely, *Shklar and Rosen (2003, p. 5)* define a web-based application as "a client/server application that uses a web browser as its client program, and performs an interactive service by connecting with servers over the Internet (or Intranet)." To emphasize the difference to traditional web sites, *Baxley (2003, p. 1)* notes that web-based applications "establish a unique session and relationship with each and every visitor" and "allow users to create, manipulate, and permanently store data". *Booch (2001, p. 2)* gives a canonical architecture of web-based applications (see Figure 1) and discusses architectural differences to traditional client-server applications.

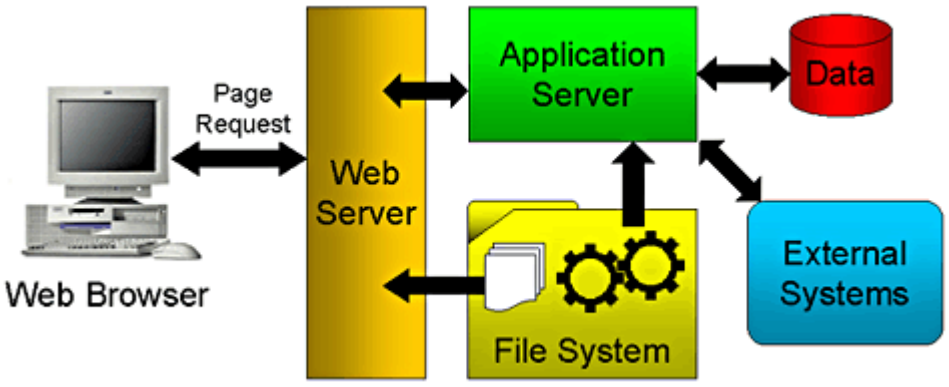


Figure 1: Canonical Web architecture *(Booch, 2001)*

To conclude this section it should be noted that in text books and articles as well as in practice the terms *web-based application* and *web application* are almost always used interchangeably.

1.1.2 Developing Web-based Applications

Web-based applications are typically developed for large numbers of users across various locations. In practise this implies the demand to be sufficiently adaptive to different users, technical environments and communication channels. In principle web-based applications can be open to arbitrary users or restricted to particular user groups. However, even in the latter case it is often unclear where, when and how users access the web-based application. Acknowledging this challenge *Bhattacharya (2006, p. 1)* states that web-based application "are the trickiest to manage and maintain" and concludes four key features that one must addressed in a good web-based application: scalability, availability, maintainability and reliability. *(Bhattacharya, 2006)* also proposes metrics for measuring the abilities of a web-based application in terms of these features and discusses best programming practises that promote these features.

To provide dynamic content web-based applications originally used common gateway interface (CGI) scripts that were requested by the web browser and executed on the computer hosting the web server. Since then a large variety of technologies for

dynamic content delivery has been proposed for web-based applications. Most popular are scripts, that is, small fragments of code that are embedded in or referenced to by web pages. Scripts are interpreted and executed when the respective web pages is requested. Again this can be done by the web browser or by the web server. Server-side scripting is for example done with PHP, JSP and ASP. Client-side scripting is for example done with JavaScript or VBScript.

To overcome common problems such as network latency, high interface complexity and slow server responsiveness, various proposals for improved client-server interaction have been introduced to provide developers more advanced capabilities in their effort to make web-based application more responsive, interactive and customizable. The most popular approach for achieving this purpose is AJAX *(Garrett, 2005)*. AJAX stands for "Asynchronous JavaScript and XML". It is entirely built on existing, well-established standards that are familiar to most web-application developers. Technologies that are recommended in the Ajax approach include HTML/XHTML, CSS, DOM, XML, XSLT, XMLHttp and JavaScript. *(Garret, 2005)* lists five key characteristics of applications built according to the AJAX approach:

- a user interface constructed with open standards such as the dynamic hypertext markup language and cascading stylesheets,
- a dynamic, interactive user experience enabled by the DOM,
- data exchange and transformation using the Extensible Markup Language (XML) and extensible stylesheet language transformations,
- asynchronous client/server communication via XMLHttpRequest, and
- JavaScript as the lingua franca joining all the components together.

The integration with JavaScript and XML proposed by AJAX, enables web-based applications to send a XML messages (small XML data fragments) describing their request instead of a complete page post-back, which generates a chunk of XML describing the matching information data as a response. This type of web application development techniques with client-side scripting language and semi-structured data format makes partial client-server data exchange more efficient than full browser and server communication which often requires rerendering the entire web page *(Smith, 2006)*. A number of commercial web sites have been built according to the AJAX approach to improve the user experience. Examples include Google Maps, Google Suggest, Gmail, but also a variety of popular blogs on the Web.

An alternative approach is to use special software components that extend the web browser or web server. Examples of web browser extensions are plugins that allow the execution of Java applets or Flash applets. Examples of web server extensions are web containers that allow the execution of Java servlets. Both client-side and server-side solutions have particular advantages and disadvantages when it comes to features such as scalability, maintainability, availability, and reliability, but also security. In practise one often finds a mixture of both client-side and server-side

technologies. For a detailed discussion, see for example *(Shklar and Rosen, 2003)* or *(Doyle and Lopes, 2005).*

The fast emergence of new powerful technologies resulted in more and more complex web-based applications. Unfortunately the rapid development of such applications often abets bad programming practises.

Generally speaking, web-based application development does not have rigour, systematic approach and quality control and assurance. Due to the complexity and sophistication of web-based application development, there is growing concern about the methods and tools in which applications are created and their quality and integrity. Since there is no disciplined process of web-based application development, developers may encounter various problems for successful development, deployment, operation and maintenance. In order to avoid such web crisis and achieve greater success in developing web-based applications, there is a pressing need for disciplined approaches and tools for development, deployment and evaluation of web-based systems, see *(Murugesan et al., 2001).*

1.1.3 New Challenges and Opportunities with Web 2.0

The toolbox of web-based application developers has undergone various reviews and changes over the last few years. Existing technologies and methods have been refined and new ones have been adopted. Often these new technologies and methods are the consequence of new approaches or paradigms that emerged in the community of web-based application developers. A recent idea that caused much attention and discussion is Web 2.0.

The intention of Web 2.0 is to bring people on the Web together in a more dynamic, interactive space. It is originally described as “an attitude not a technology” *(Davis, 2005)*. The truth is that Web 2.0 is a difficult term to define, even for web experts. The core idea is about enabling and encouraging user participation through open applications and services. The Web 2.0 movement combines a variety of new approaches for integrating services, application and their underlying databases that communicate with one another, exchange data and adapt to individual users. *O'Reilly (2005)* attempted to outline the concepts behind Web 2.0, see .

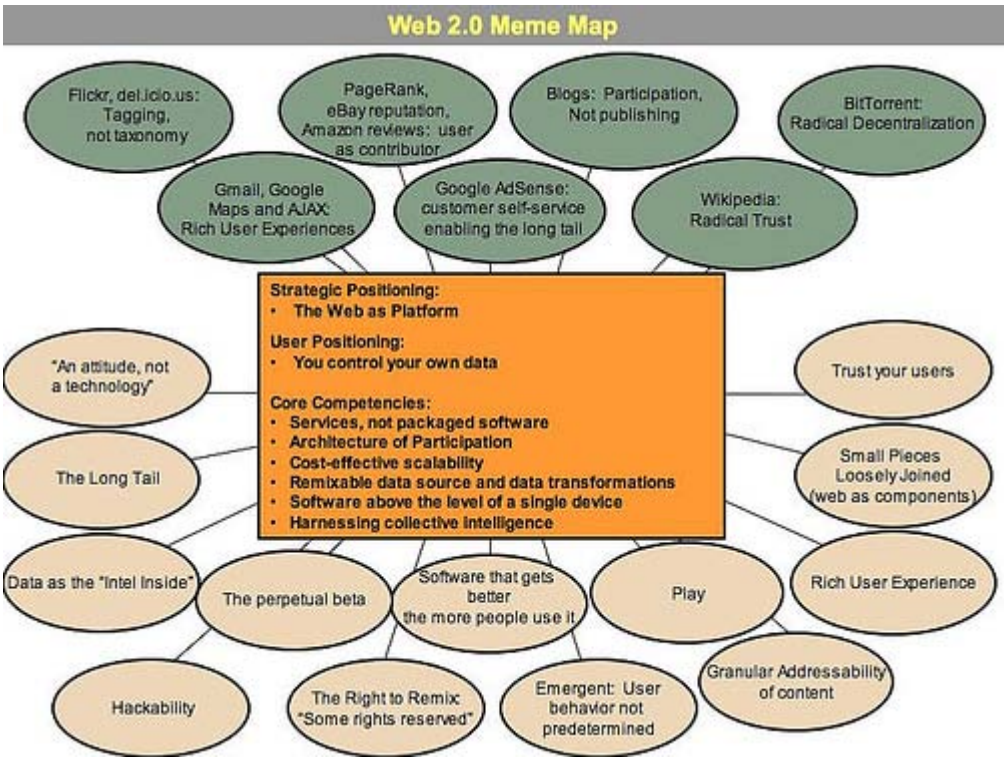


Figure 2: Web 2.0 'meme map' ***(O'Reilly, 2005)***

O'Reilly (2005a) offers a widely adopted definition that regards Web 2.0 applications as "those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a form that allows remixing by others, creating network effects through an architecture of participation, and going beyond the page metaphor of Web 1.0 to deliver rich user experiences".

Clearly, such Web 2.0 web-based applications require interactive user interface and effective model for application integration and deployment. The new Web 2.0 idea focuses on the benefits of easy to use and performance of web-based applications. A Web 2.0 application should promise to expand the functionality of a core business application, knit together multiple services and deliver a feature-rich user interface to enhance the customer experience and employee productivity. In order to fulfil those requirements along with application development processes, a variety of web-based application development tools have been released to make development process more effective and efficient.

The recent emergence of Web 2.0 applications results in new challenges for web-based application development. At the time of writing, many of the most promising Web 2.0 applications are being published continuously. The future development of these Web 2.0 applications requires adopting these technologies and fulfilling the goals of applications. It also results in a number of fundamental changes in design patterns and programming models. *O'Reilly (2005)* listed core changes for

business model of Web 2.0 applications:

- Operations must become a core competency
- Users must be treated as co-developers

These changes of the business model have a number of significant consequences for the development of web-based applications. In particular the increasing availability of web services, leads more and more businesses to look for published web services that can be called from inside their applications. This approach calls for highly reliable programming environments that are able to communicate with and make use of a variety of heterogeneous services, applications, and components on the Web. *(O'Reilly, 2005)* mentions the following key criteria:

- support lightweight programming models that allow for loosely coupled systems,
- think syndication, not coordination, and
- design for hackability and remixability.

These criteria are crucial for the successful development of Web 2.0 applications Web-based application developers, both experienced and novice, are trying to find easy-to-use, flexible development tools that suits them best in developing web-based applications that meet these criteria. In my investigations of typical web-based application tools these key criteria played a central role.

1.2 Web-based Application Development Tools

Generally speaking, web-based application development does not have rigour, systematic approach and quality control and assurance. Due to the complexity and sophistication of web-based application development, there is growing concern about the methods and tools in which applications are created and their quality and integrity. Since there is no disciplined process of web-based application development, developers may encounter various problems for successful development, deployment, operation and maintenance. In order to avoid such web crisis and achieve greater success in developing web-based applications, there is a pressing need for disciplined approaches and tools for development, deployment and evaluation of web-based systems *(Murugesan et al., 2001)*. This research paper explores different types of web-based application development tools and assesses potential problems during development process under new developing environment. It is expected is to find out possible workarounds and solutions to handle these problems.

A variety of web-based application development tools are available in industry. They all have their specific abilities and features, which become their own advantages and disadvantages. Therefore, comparing these tools becomes significant to the success and satisfaction of a web-based application development.

A recent survey paper *(Doyle and Lopes, 2008)* of technologies for web-based application development provides a state-of-the-art comparison of development approaches based on these web technologies. The following table shows how they classify the types of development approaches and technologies used for each type.

Technology	Exemplars	Key Properties and Common Usage
Frameworks		
Scripting language-based	Twisted (Python), WebWare (Python), Snakelets (Python), Rails (Ruby)	Portable. Small to large-scale applications with complex navigation requirements.
Application-driven	Struts, Spring MVC, WebWork, Maverick, Barracuda	Portable. Scalable. Enterprise web sites with complex navigation requirements.
Page-driven, component-based, desktop model	Echo, wingS, WebCream, Wi.Ser	Portable. Complex. Ease transition to Web programming.
Page-driven component-based, Web model	WebObjects, Tapestry, ASP.NET WebForms	Portable. Scalable. High usability. Rapid development. Simple to medium-complexity navigation requirements.
Application/Page Hybrid-driven component-based	JavaServer Faces	Portable. Scalable. High usability. Enterprise web sites with complex navigation requirements. Rapid development.
Portal composition	Java Portlet specification, WebParts, Tiles, SiteMesh	Portable. Scalable. Enterprise portal development.
Model Driven Development		
Data-centered	AutoWeb, RMM, OOHDM, Oracle Web Development Suite, WebML, WebRatio	Data-intensive applications.
GUI-centered	CodeCharge, CodeSmith, DeKlarit, Fabrique	Interation-intensive applications.
MDA-compliant	Oracle ADF, IBM/Rational Rapid Developer	OMG standard.
Development Environments		
Authoring tools	Adobe PageMill, Amaya, Microsoft Office (Word, Excel, PowerPoint)	Static content creation and editing.
Web site management	FrontPage, NetObjects Fusion, MacroMedia Dreamweaver	Mostly static web sites with simple navigation requirements.
Full-fledged environments	Microsoft VisualStudio.NET, Sun Java Studio Creator, WebSphere Application Developer	Programming-intensive.
Others		
Programming languages	MAWL, <bigwig>, JWIG, functional continuations, Cocoon Flow	Research projects.

Figure 3: Development approaches and technologies *(Doyle and Lopes, 2008)*

Following the classification of web-application development tools provided in *(Doyle and Lopes, 2008)*, see Figure 3, we divide them into three classified types for better investigation and comparison. They are model-driven development tool (e.g. Oracle APEX), integrated development environment packages (e.g. ActiveGrid/WaveMaker

Development Environment) and development framework on pure programming language (e.g. Ruby on Rails). Although they are not complete, they can briefly represent typical existing tools in the industry. This research paper focuses on those three types of development tools to deal with XMLType datatype in the following sections.

1.2.1 Oracle Application Express (APEX)

The first type of web-based application development tools are model-driven development tools. The typical example is Oracle APEX. Several Oracle database distributions such as Oracle Database 10g Express Edition have APEX included as the prime tool for creating data-centric web-based applications. Oracle Application Express (Oracle APEX) was formerly called HTML DB, which is a rapid web application development tool for Oracle database. Oracle refers to HTML DB as a "declarative development tools" *(Klaene, 2004)*. It is especially designed to meet simple application development requirements. This platform provides a variety of wizards that allow end users to operate on database tables, build forms and reports upon Oracle databases easily. It has become a promising web-based application development solution based on Oracle database.

According to Oracle's Developer's Guide for the Oracle DBMS and APEX *(Baker, Romano, and Winters, 2008)* APEX is divided into the following four major components:

- Application Builder – Securely assemble web pages and business rules into applications
- SQL Workshop – Create and manage database objects and use the Graphical Query Builder to create SQL queries – even if you have no knowledge of SQL
- Utilities – Import and export data from the database, generate DDL, view object reports, and restore dropped database objects
- Administration – Manage preferences, users, and view application reports

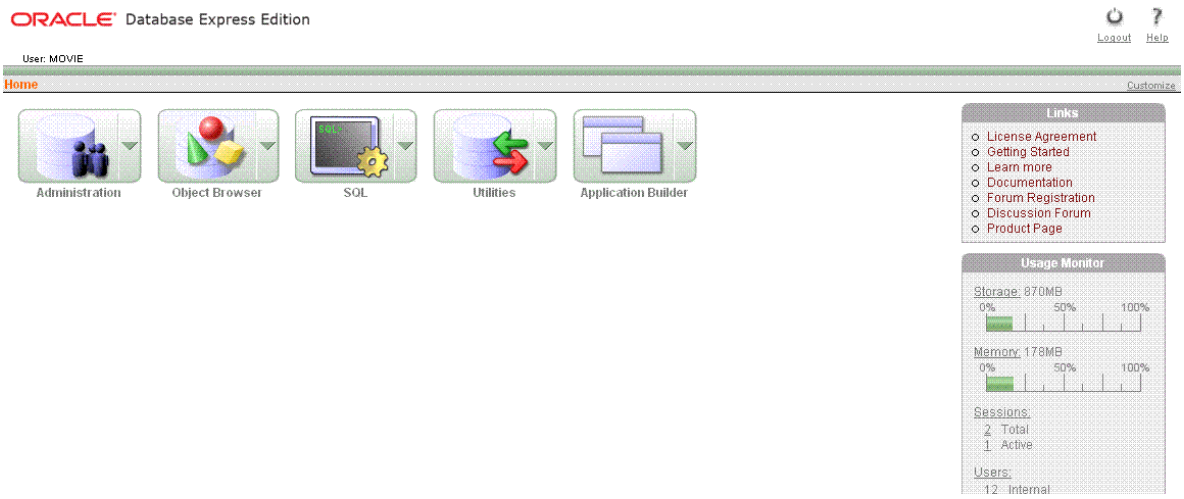


Figure 4: Screenshot of Oracle APEX home page

The Oracle APEX provides a metadata repository where definitions of applications and APEX engine are stored. Similar with most web applications, the application express engine is accessed from a web browser through the HTTP Server (Apache). The application express engine is embedded in Oracle database. Within Oracle Database 11.1 or higher or Oracle Database 10g Express Edition, you can also remove Oracle HTTP Server (Apache) from the architecture and replace it with the embedded PL/SQL gateway.

A remarkable difference between an Oracle APEX application and most other web applications is that it stores user session data or metadata into its own database rather than in an in-memory session object for each user. It enables rapid development by means of wizards and pre-built components. Furthermore, developers are capable of extending the functionality of an application with custom PL/SQL or JavaScript. The application builder, for example, allows developers to implement customized logic using PL/SQL or JavaScript or to call existing PL/SQL functions and procedures as well as JavaScript functions from an existing JavaScript library. This supports reuse and modularization during the application development process.

1.2.2 WaveMaker (formerly ActiveGrid)

The second type of web-based application development tools are integrated development environment packages. ActiveGrid Enterprise Web 2.0 is a typical example of this type. It is intended to achieve short work of integrating existing services, applications and databases into rich, integrated Web 2.0 applications *(ActiveGrid, 2006)*. ActiveGrid Studio and ActiveGrid Server are designed for developer to deliver rich, interactive Web 2.0 web applications. As claimed by ActiveGrid, installing ActiveGrid Studio takes less than 15 minutes. It's delivered with everything developers need, including an integrated web server, embedded database and sample Web 2.0 applications. Recently (November 28, 2007) in San Francisco, ActiveGrid announced that it would change its name to WaveMaker, and released version 3.0 of its now renamed WaveMaker Rapid Deployment Framework

the same day, together with a new product, called WaveMaker Visual Assembly Studio 3.0, see *(Constantin, 2007)*. On this occasion the CEO of WaveMaker, Christopher Keene, emphasised that "they created WaveMaker's new flagship product line to simplify the development process, accelerate assembly and deployment time, and dramatically improve business productivity—all directly impacting an enterprise's bottom line", see *(Handy, 2007)*. The key point of both ActiveGrid and WaveMaker is to provide developers easy-to-use, data-driven, visual tool so that developers can build scalable web-based applications which meet CIO requirements.

The chief components of ActiveGrid Studio and WaveMaker Studio are somehow changed. The new WaveMaker studio is a pure web-based application that can run on most web browsers while ActiveGrid studio was a standalone application development environment. In order to make the paper up-to-date, we focus on WaveMaker studio as application tool to investigate. The key parts of WaveMaker are editor tabs which cover of the main components of WaveMaker, see *(WaveMaker, 2007)*, including:

- Project dashboard: a user friendly overview of the project application including data objects and services operations and data model.
- Page designer: main editor where developers use to build your application pages.
- Service: allows you to edit existing services calls and add custom services calls
- Data model: the place where all imported database information (tables, data types, values…) are listed and developers can import new data model from supported databases.
- Security: shows authentication relative information about the web-based application.
- Source: the area where all types of source codes are listed including script, CSS, markup, widgets and application source codes.

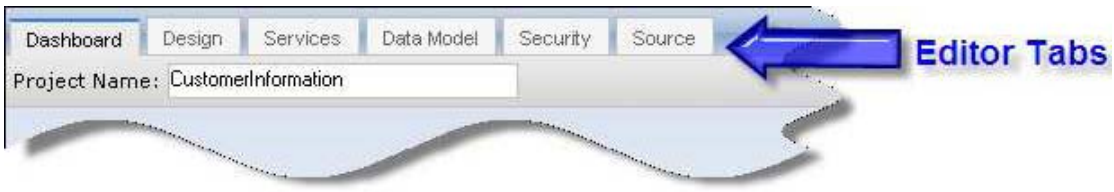


Figure 3: WaveMaker editor tabs

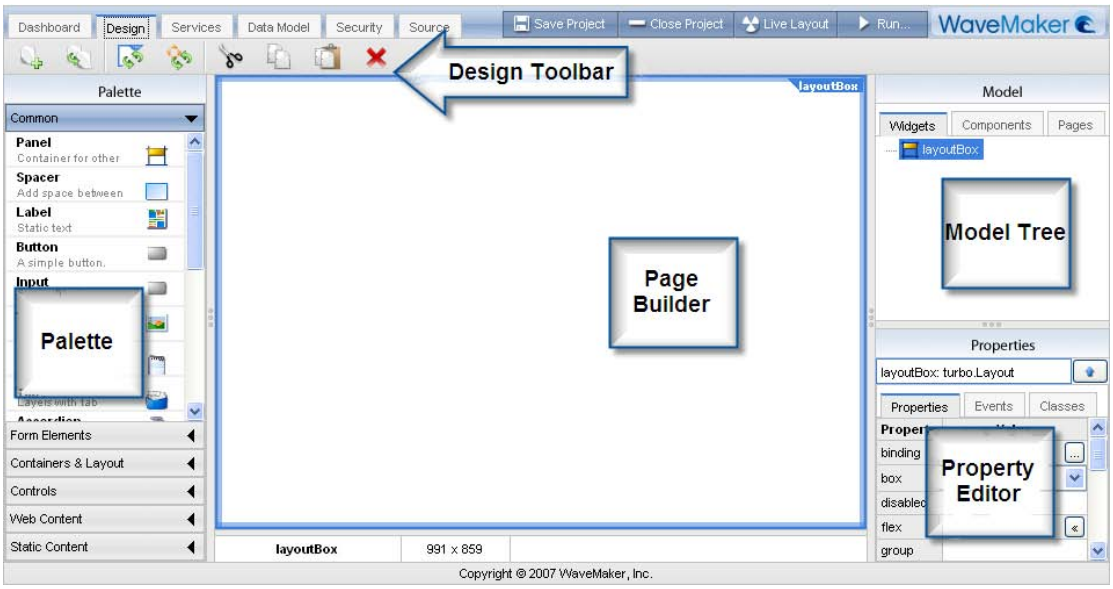


Figure 5: Page Designer editor

Both ActiveGrid and WaveMaker provide developer different types of databases support. WaveMaker supports most of the popular databases such as MySQL, Oracle and SQL Server. By default, the installation also includes Hypersonic SQL DB (HSQLDB), as an embedded database. This database is included for demo and testing purposes. The data model is created as the data source used for web-based application development. All those data models are generated by importing external databases. WaveMaker provides user-friendly integrated development environment where services and operations can be easily generated by creating service components. Furthermore, both ActiveGrid and WaveMaker have alternative methods or script languages for creating custom services and operations. ActiveGird supports three types of scripting languages for defining custom services which are Python, PHP and Java. WaveMaker, however, reduced the number of choices to two which are Java and JavaScript. Whatever methods or languages those two web-based application tools provide; developers can create their own custom services and add service call components into the application. It enhances flexibility for this type of development tool when developing web-based applications for different levels of developers.

1.2.3 Ruby on Rails

The third type of web-based application tool we study are frameworks based on pure programming languages. Typical examples are Java Struts and Ruby on Rails. The underlying programming languages (Java, Ruby) provide essential functionalities for web-based application development. With the development of those object-oriented scripting languages, those languages have more and more strengths including ease of use, short development time, and high performance. Moreover, those programming languages provide several built-in class libraries which are easy to be extended for putting the promise of reuse into immediate practice. Developers get the data source for a specific web-based application. Scripting languages also can connect with corresponding popular databases and retrieve data from database tables. They usually

have specific libraries to access corresponding databases through database APIs. With the great demanding of database-driven web-based application development, certain scripting language development environment framework is introduced.

Rails is a full-stack framework for developing database-backed web-based applications according to the Model-View-Control pattern. Rails provides an integrated development environment for creating web-based application with Ruby. Ruby on Rails is the programming language framework for web-based application development I focus on in this paper. This framework was first released by David Heinemeier Hansson in July 2004. It is an offspring of his work on the web-based project management application Basecamp (for a web application company called 37 Signals). While developing Basecamp, Hansson realized that a lot of the code he was writing could be extracted into a framework that could be used as part of other future application, see *(Herrán, 2006)* or *(Williams, 2007)*. That makes the design and development of web-based applications using Ruby easier based the MVC (Model-View-Control) architecture. It is expected to make the maintenance of the applications simpler than other frameworks. The Ruby on Rails' MVC architecture is used for web-based applications giving like a result frameworks. It shows:

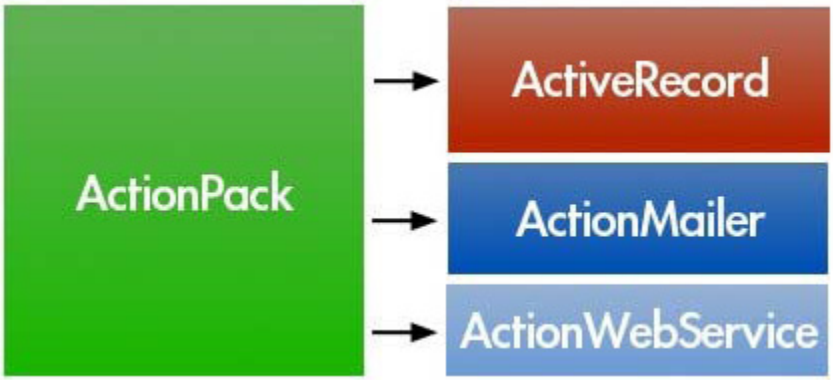


Figure 6: High level component/service view

- ActiveRecord: Rails M (Model) support which follows the standard Object/Relational Model (ORM) to communicate with the database.
- ActionPack: Rails VC (View-Control) which handles dispatched requests, controllers and views. It contains ActionController and ActionView modules within it.
- ActionMailer: A component that delivers and receives email.
- ActionWebService: ActionWebService supports SOAP and XML-RPC web services.

Rails serves as an add-on to the Ruby programming language. This add-on contains a library that is full of Ruby code, scripts for generating components of web-based applications. It provides Ruby more strengths and flexibilities for web-based application development including database connection and web services. Developers

can individually consider implementations of “Model” part and “View” part which enhance the reusability of the web-based application developed under Ruby on Rails. Rails also supports a wide range of databases, including all common production database platforms. Several different RDBMS systems are supported including MySQL, PostgreSQL, SQLite, IBM DB2, Oracle and Microsoft SQL Server. That makes the web-based application development more flexible on data source. Ruby, as a promising programming language, also makes development a positive strength on this Rails framework. After all, Ruby on Rails is a very good framework to start programming web applications because of the simplicity of learning about it and its efficiency.

2 Dynamic Content from XML Data

2.1 XML

This research is motivated by the wide-spread acceptance of XML (the Extensible Markup Language) for data exchange, data modelling and data integration (XML, 2006). XML and related concepts are maintained by the World Wide Web Consortium (W3C) that develops and regularly releases standards (called W3C recommendations) that have been widely adopted by the industry. It was originally designed to be a markup language for flexible presentation of data in web environments. Meanwhile, XML has emerged as a general format for data exchange between multiple heterogeneous applications over the Internet. It has proved to be a useful and simple enough medium for generating, exchanging and retrieving data. Its usage has greatly expanded beyond its initial purpose and reached a wide range of implementations in a many (commercial and free) database tools as well as a huge number of web-based applications.

Regarding the basic difference between XML and relational data, *(Chamberlin, 2002)* emphasises that "XML data are different from relational data in several important respects that influence the design of a query language. Relational data tend to have a regular structure, which allows the descriptive meta-data for these data to be stored in a separate catalog. XML data, in contrast, are often quite heterogeneous, and distribute their meta-data throughout the document. XML documents often contain many levels of nested elements, whereas relational data are "flat." XML documents have an intrinsic order, whereas relational data are unordered except where an ordering can be derived from data values. Relational data are usually "dense" (nearly every column has a value), and relational systems often represent missing information by a special null value. XML data, in contrast, are often "sparse" and can represent missing information simply by the absence of an element. For these and other reasons, existing relational query languages are not directly suitable for querying XML data".

XML	**Relational Data Model**
Data in single hierarchical structure	Data in multiple tables
Nodes have element and/or attribute values	Cells have a single value
Elements can be nested	Atomic cell values
Elements are ordered	Row/column order not defined
Elements can be recursive	Little support for recursive elements
Schema optional	Schema required
Direct storage/retrieval of XML documents	Joins often necessary to retrieve data

Query with XML standards (XQuery, XPath)	Query with SQL

Figure 7: Comparison of XML and Relational Data

```
<Result>
    <Customer>Sebastian</Customer>
    <Item>
        <ID ItemId="01"/>
        <Title>Toyota Car</Title>
    </Item>
    <Bids>
        <Bid>
            <Bidder>
                <Name BidderName="Yi"/>
                <Email>y.chen.1@massey.ac.nz</Email>
            </Bidder>
            <Kind>Flat</Kind>
            <Amount FixAmount="50"/>
        </Bid>
        <Bid>
            <Bidder>
                <Name BidderName="Dagong"/>
                <Email>D.D.Dong@massey.ac.nz</Email>
            </Bidder>
            <Kind>Range</Kind>
            <Amount MaxAmount="200"/>
        </Bid>
    </Bids>
</Result>
```

Figure 8: An XML data fragment with auction bid data

Due to the high flexibility of XML format, it can be used to present many kinds of information. For instance, the biologist generating DNA sequences data may make it available for other biological researchers using XML format stored in XML databases. As it is becoming more and more important, a variety of standards have come up around it, most of which are defined by the World Wide Web Consortium (W3C) such as XML Schema *(XML Schema, 2004)*, XPath *(XPath, 2007)*, XQuery *(XQuery, 2007)*, and XSLT *(XSLT, 2007)*. XML makes it possible to exchange data with a united format on applications regardless of storage dependence. Currently, possible persistent XML stores includes native XML database, XML files on a file system, or XML stored in SQL databases. Since XML has already been regarded as a standard for data exchange, it is obvious that it could be another type of data stored in traditional relational databases like other popular data types. Actually, there are a variety of popular relational databases which already provide XML data type support. In the

next section, we will investigate different types of XML databases and briefly look into some typical XML data support databases which could be used as back-end data storage for potential web-based application development.

2.2 Processing XML Data with DBMS

Nowadays, more and more business applications gradually regard XML as a universal medium. With the greatly increasing popularity of XML, it is natural that enormous XML data storage will merge. XML data can be stored into different kinds of persistent XML stores. Basically it is divided into different categories based on different methods that are native XML databases for document-centric applications and XML-enabled databases for data-centric applications or hybrid XML database for both types of applications.

2.2.1 Native XML Databases

According to *(Liotta and Preimesberger, 2003)* a native XML database (NXD) first defines a model for an XML document and then stores and retrieves the document according to the model. The idea of native XML databases is based on is to use a specific data model that can store any XML document. This specific data model is a document-centric model which exploits a native XML database to store a huge number of XML documents in repositories for information retrieval. *Bourret (2005)* emphasises that “applications use document-centric documents in a variety of ways, but most uses falls into four broad categories: managing documents, finding documents, retrieving information, and reusing content”. Some developers prefer a native XML database as it offers more flexibility with storing, retrieving, and processing persistent data in the form of XML documents. It may understand the XML documents structure better and maintain the original data hierarchy. *Staken (2001)* points out that native XML databases “store XML documents as a unit and will create a model that is closely aligned with XML or one of XML's related technologies like the Infoset or DOM”. This model includes arbitrary levels of nesting and complexity, as well as complete support for mixed content and semi-structured data. This model is automatically mapped by the NXD into the underlying storage mechanism. The mapping used will insure that the XML specific model of the data is maintained”. Timber, eXist, and also Tamino are examples of native XML database management systems that are currently in use. They have been developed for the purpose of storing XML data directly with keeping the tree structure.

2.2.2 XML-Enabled Databases

Another alternative approach to store XML data is using XML-enabled databases. It is normally used for data-centric applications. According to *Bourret (2005)* XML-enabled databases are “databases with extensions for transferring data between XML documents and themselves, primarily for data-centric applications”. That means XML-enabled databases are normally established based on another specific data model that is data-centric model. It is completely different from document-centric

model mentioned before for native XML databases. As *Obasanjo (2001)* expressed the data-centric model that "in a data-centric model where data is stored in a relational database or similar repository; one may want to extract data from a database as XML, store XML into a database or both. For situations where one only needs to extract XML from the database one may use a middleware application or component that retrieves data from the database and returns it as XML". It describes the data structure used in XML-enabled databases which typically resides in relational databases. Those databases are supposed to serve as not only a central repository of XML-documents and DTD schema definition but also provide all necessary data management functionalities. Relational database system was proposed to satisfy those requirements by "devising ways to 'shred' XML documents into relations, and translate XML queries into SQL queries over these relations" *(Tatarinov et al., 2002)*. They simply map all XML to a traditional database. DB2 XML extender, Oracle 9i/10g and SQL server 2000 are some typical sample database systems which are all integrate XML data processing into the relational databases even though the approaches are slightly different.

2.2.3 Hybrid Databases

In some cases, both document-centric and data-centric models of XML usage could be used within the XML storage. The best data storage choice is usually considered to be native XML databases. However, the lines that many see between XML-enabled databases and native XML databases are becoming blurred. In the IBM DB2 documentation *(Beyer and Cochrane, 2006)* three key reasons are discussed that call for hybrid database systems that can handle both relational and native XML data:

- XML and relational data coexist and complement each other in enterprise solution.
- A successful XML repository requires much of the same infrastructure that already exists in a relational database management system.
- XML query languages have considerable conceptual and functional overlap with SQL.

Tamino was originally built as a native XML database management system, but can now be regarded as a hybrid system. Thanks to its SQL engine, it provides means for storing and retrieving both kinds of data as well as for retrieving them with external data sources and applications.

The major relational database systems have been providing some XML support for several years, predominantly by mapping XML to existing concepts such as CLOBs or relational and object-relational tables. The limitations of this approach are widely known in research and industry. In particular after the release of the W3C recommendation of XQuery *(XQuery, 2007)*, a new generation of database management systems has been released that provide comprehensive native XML

support. Version 9 of IBM's DB2 database management system is probably the most prominent example. "Native" means that XML documents are stored on disk pages in tree structures matching the XML data model. This avoids the mapping between XML and relational structures, and the corresponding limitations. That makes these DBMS true hybrid systems which places equal weight on XML and relational data management, see *(Nicola and Van der Linden, 2005)*.

2.2.4 Comparison

A detailed comparison of XML-enabled and native XML databases is given in *(Thuraisingham, 2002)*, see also *(Bansal and Alam, 2001)*. For XML-enabled databases one has the following advantages:

- Once configured they are fast since most data processing is done by the DBMS.
- Applications can use the functionality provided by DBMS extensions such as fuzzy searches, synonym searches, and searches by sentence or paragraph etc, and it is easy to add XML support to existing applications that already use the same backend database.

On the other hand, one faces the following disadvantages:

- Without adequate support of the DBMS a considerable amount of programming is needed before one can actually store, retrieve, or query XML documents.
- Developers use proprietary extensions to SQL (provide by the respective DBMS) rather than XML or other standards to perform non-trivial operations and manipulations on the XML data.
- As the database stores the document as a whole including tags etc., it consumes more space in the DBMS.

In contrast, native XML databases observe the following advantages:

- The DBMS provides comprehensive support for the management of XML documents, including storage, access and search (XPath, XQuery), specialized functions, etc.
- There is no performance penalty during the runtime of the database caused by the translation of data structures to and from XML.
- Developers save time with programming. In particular they are not concerned with non-XML data structures (like tables and transformations from and to XML) when the XML documents have to be stored.
- A change in the XML schema may not require a corresponding change in the database schema and mapping. Schema independence guarantees high flexibility and makes application development easier.
- They further provide a good temporary storage place for message queuing and other applications in which data needs to reside briefly in order to be queried or

reordered before being retransmitted in XML format to their final destination.

- Querying in an unstructured document is easier than in a relational database.

On the other hand, there are some disadvantages:

- For the time being they need to be considered as not less reliable as they are not yet mature, but still under development. New methods need to undergo extensive testing (as done for relational data for ages).
- Existing applications need to deal with yet another database.
- Due to the heterogeneous nature of XML they provide a lower level of data integrity, such that it may be difficult to build standard database instances.

Some of the disadvantages of native XML databases may disappear over time due to the ongoing efforts of the suppliers of database tools and DBMS. Currently, however, XML-enabled databases are often the only available choice for developers. Though native XML databases have the potential to save developers valuable time with developing and testing, almost all current web-based application development tools do not have database support on this type of XML databases.

For this thesis several web-based application development tools with a backend database that supports XML have been investigated in order to find out an efficient method for those tools to deal with XML data residing in the database. For fair comparison, an identical database has been chosen and implemented such that it can be supported by all investigated web-based application development tools. Before describing this database in more details I will briefly introduce the XMLType data type defined in the Oracle 10g DBMS, see *(Oracle, 2005a)*.

2.2.5 Oracle's XMLType

Since XML data becomes more and more popular as an information exchanging media, a variety of relational databases have already announced and released their latest version databases which allow database users and developers store XML data or document into them. In order to achieve efficient XML data storage, a new data type XMLType is introduced for storing XML data. As defined from Oracle, XMLType is a native server data type that allows the database to understand that a column or table contains XML *(Oracle, 2005a)*. It is similar to the way that normal popular data types. End-users can use the XMLType data type like other data type such as creating a XMLType column, declaring PS/SQL variables or defining and calling PS/SQL procedures and functions. Since XMLType is also an object type, end-users can also create a table of XMLType. By default, an XMLType table or column can contain any well-formed XML document.

2.3 Support for XMLType in Web-based Application Development Tools

After introducing new data type for information exchanging – XML and XML supported databases, we are going to investigate how recent web-based application development tools deal with this new data type stored in database and what are the potential problems from those implementations. In this implementation, we will try to use all three types of web-based application development tools to create web application on a simple Oracle database table with XMLType column. The table structure is as follows:

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(4,0)	No	-	1
NAME	VARCHAR2(30)	Yes	-	-
CONTENT	XMLTYPE	Yes	-	-
				1 - 3

Figure 9: A table definition with an XMLType column

This table is mainly defined to store some XML documents. These XML documents contain data about bank account information. The "content" of an XMLType column is declared to hold XML data from these XML documents. All account related XML documents are inserted into this database and ready for application development.

ID	NAME	CONTENT
1	acct1.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>0</ACC_NO> <CST_ID>0</CST_ID> <ACC_BALANCE>1040</ACC_BALANCE> <ACC_CREDITLIMIT>3.14159</ACC_CREDITLIMIT> <ACC_CREATEDATE>1967-08-13</ACC_CREATEDATE> <ACC_CARDTYPE>MASTER</ACC_CARDTYPE> <ACC_ENABLED>true</ACC_ENABLED> </ACCOUNT>
2	acct2.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>7</ACC_NO> <CST_ID>0</CST_ID> <ACC_BALANCE>1040</ACC_BALANCE> <ACC_CREDITLIMIT>34.14159</ACC_CREDITLIMIT> <ACC_CREATEDATE>1977-08-13</ACC_CREATEDATE> <ACC_CARDTYPE>VISA</ACC_CARDTYPE> <ACC_ENABLED>true</ACC_ENABLED> </ACCOUNT>
3	acct3.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>8</ACC_NO> <CST_ID>8</CST_ID> <ACC_BALANCE>888</ACC_BALANCE> <ACC_CREDITLIMIT>8.88888</ACC_CREDITLIMIT> <ACC_CREATEDATE>8888-88-88</ACC_CREATEDATE> <ACC_CARDTYPE>test</ACC_CARDTYPE> <ACC_ENABLED>test</ACC_ENABLED> </ACCOUNT>

Figure 10: A table with XMLType data

The main purpose of this application development investigation is exactly to observe how web-based application development tools deal with this XMLType column and what potential problems may occur.

2.3.1 Oracle Application Express (APEX)

Oracle APEX, as an Oracle integrated primary tool for managing the database and building web-based applications, has inborn advantages which seems to be possible

for giving it enough supports on XMLType data. Since Oracle APEX directly use Oracle database as data source, there is no problems to import database for developing web application. However, it also reveals unavoidable problem when developers attempts to retrieve XML data from XMLType column for listing records. Oracle APEX is able to generate an application with basic functions by using wizards as follow:

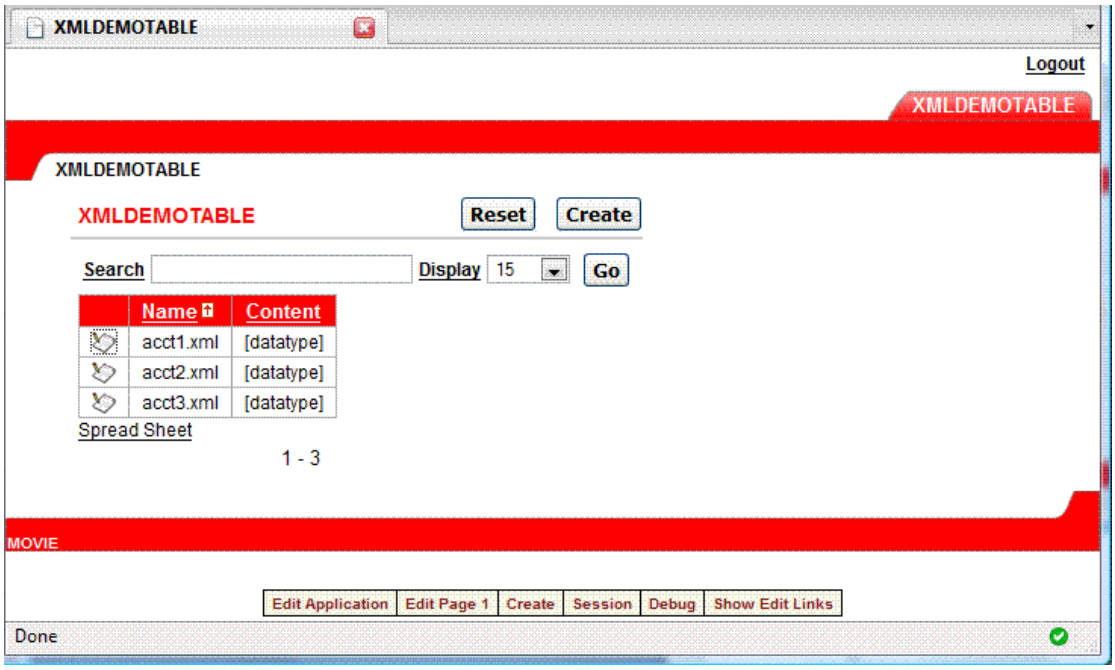


Figure 11: XML demo page in Oracle APEX

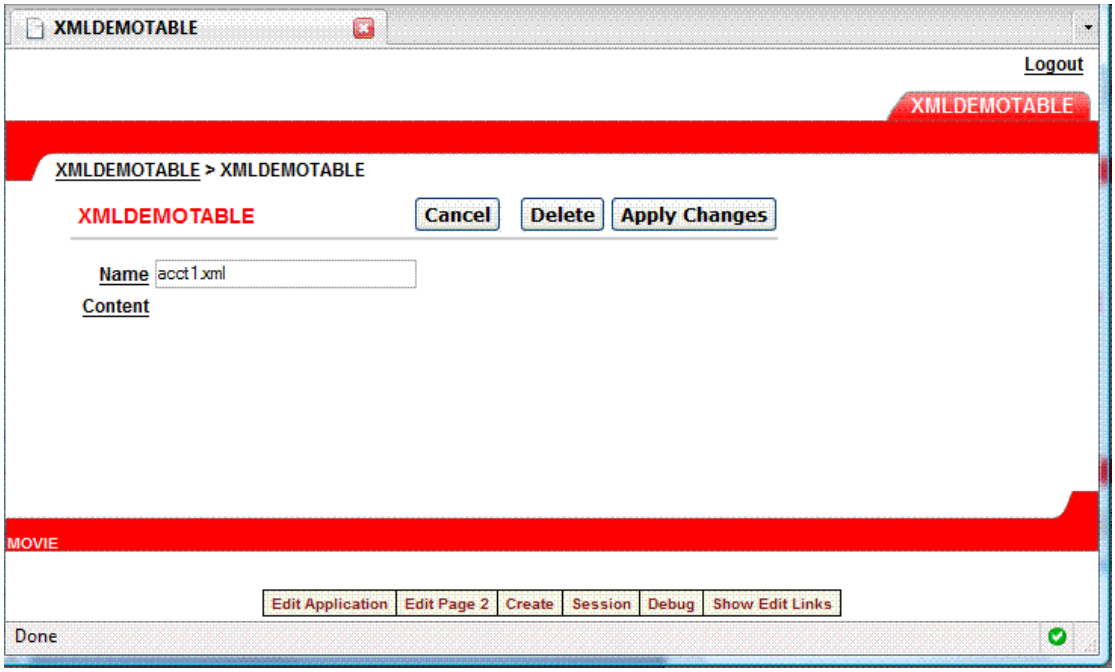


Figure 12: XML demo form in Oracle APEX

As we can see, the wizard generated forms that use automated DML are not

XMLType-aware. In the table record listing page, all “content” column data are represented as “[datatype]” rather than the XML data from XML documents. When end-users enter wizard-generated insert/update form, XML document content is still not shown for data manipulation. Although we can retrieve XMLType data from Oracle database through simple SQL script, auto-generated web pages created in Oracle APEX cannot show XMLType data value within corresponding page element. That makes it hard to develop a web application by using Oracle APEX when there is XMLType data in back-end XML-enabled database without adding certain XML-specific retrieval methodologies.

2.3.2 WaveMaker (formerly ActiveGrid)

WaveMaker has different methodology for dealing with back-end data model. Developers have to import external databases and save it as the data model for application development. As mentioned above, almost all popular databases are supported by WaveMaker including HSQLDB, MySQL, Oracle and SQL Server. In this case, we still import the Oracle database with the “Account” table storing XML documents. WaveMaker studio translates Oracle database table schema and save as a data model based on WaveMaker conventions. The final data model imported in WaveMaker is as follows:

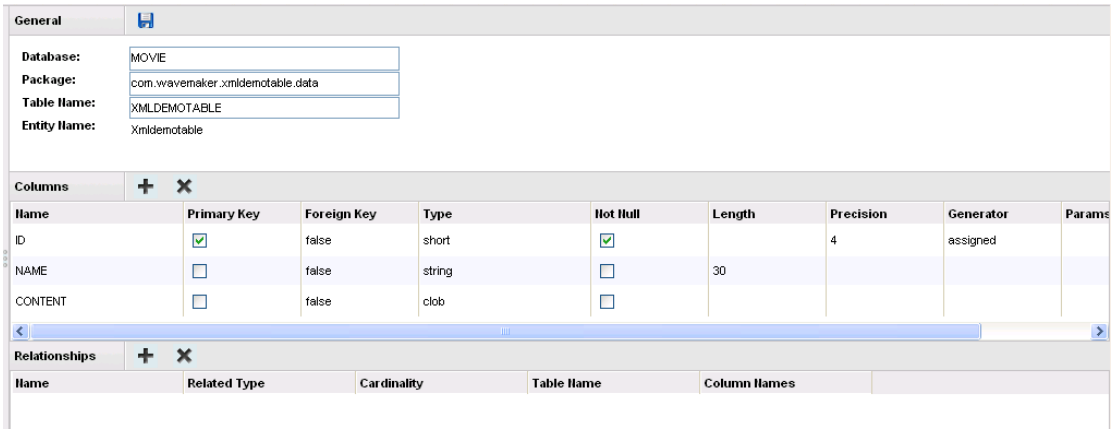


Figure 13: XML demo table structure in WaveMaker

From the imported data model definition, we can see WaveMakers has its own conventions which follows OO programming variable types such as short type for number type and string type for varchar type in Oracle database. The column with XMLType data is treated as a CLOB (Character Large Object) in WaveMakers. Since WaveMaker is mainly programmed Java, by default drivers implement a CLOB object using an SQL locator (CLOB), which means that a CLOB object contains a logical pointer to the SQL CLOB data rather than the data itself. A CLOB object is valid for the duration of the transaction in which it was created.

With the developing mechanisms, we expect to get a simple search function page on table. A test page with search function on person information table is created as follows:

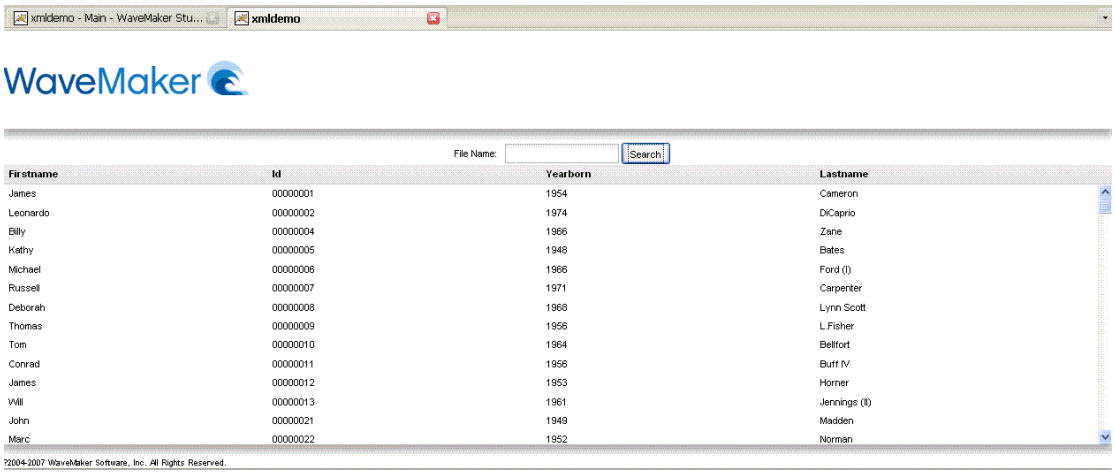


Figure 14: Sample XML demo table in WaveMaker

However, when we attempt to search the table with account related XMLType data column and show account information, the auto-generated search page cannot display table data on the page and shows a rendering exception error:

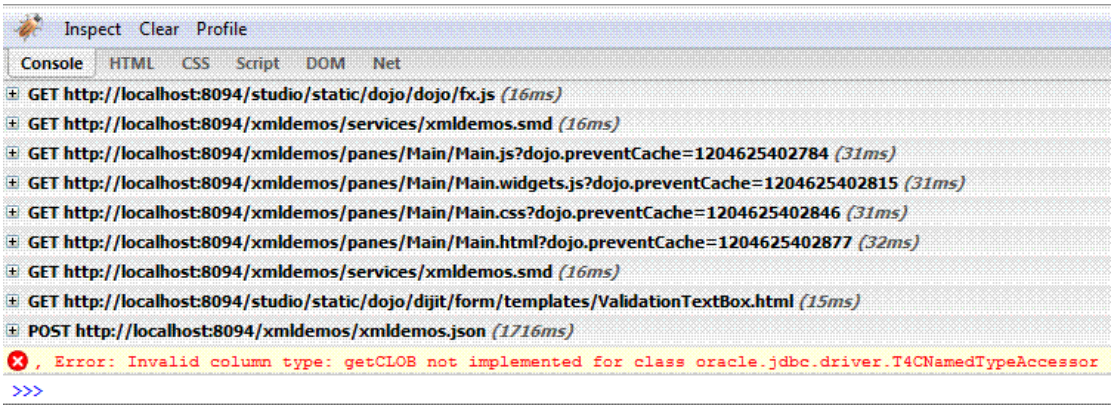


Figure 15: XML demo table exception error in WaveMaker

The same problem happens for ActiveGrid which shows exception error of unsupported data type. Both of them show that XMLType data is not a natively supported data type in WaveMaker. It is mainly because this generation of JDBC driver does not support the JDBC XMLType or XML object. When developers want to retrieve XMLType data from XML supported databases, they also have to write vendor specific code and find out another methodology or workaround to handle this data type.

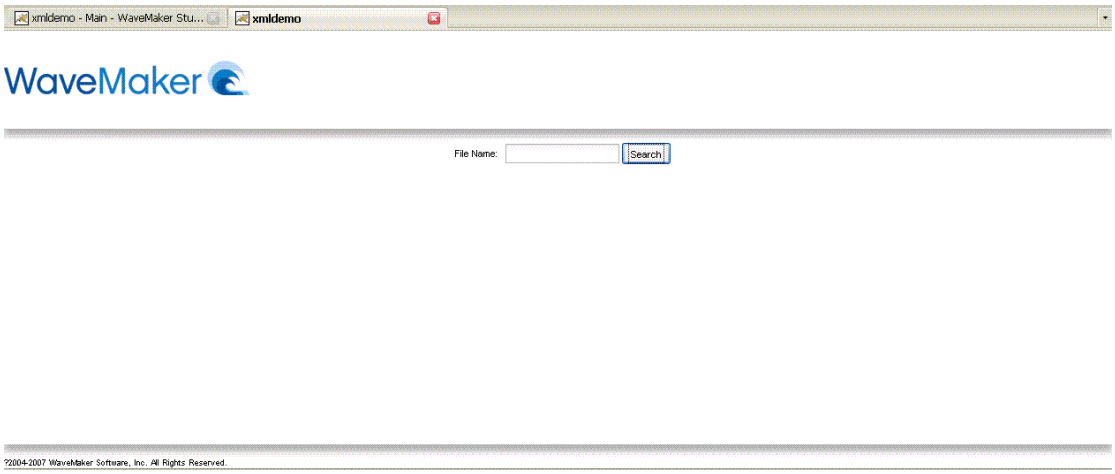


Figure 16: XML demo page in WaveMaker

2.3.3 Ruby on Rails

Unlike Oracle APEX and WaveMaker, Ruby on Rails framework shows a clear structure of the whole application. Developers have the flexibility to work on Ruby on Rails including database connection driver. Ruby on Rails support Oracle database with the ruby-oci8 connection driver, see *(Kubo, 2008)*. With the help of RubyGems, a Ruby application that allows you to install Ruby frameworks very easily, developers are able to install ruby-oci8 Oracle database adapter with it. Besides, developers also have to modify database configuration file (database.yml) in order to make the connection work. However, ruby-oci8 connection driver does not support XMLType natively. It results in exception error of unsupported data type and cannot generate web application to handle "Account" table with XMLType data type.

2.4 Research Question

Recent web-based application development tools are all mainly developed to build database-centric web application on popular relational database(s). They all provide own mechanism to handle back-end database connection in order to generate data model used for developing web application. After introducing new XMLType data type into XML-enabled relational databases, some popular databases such as Oracle, DB2, and SQL Server have already been capable of handing XML data and document within relational databases. Although these XML-enabled relational databases are ready for developers to make use of XML data or XML document, those web application development tools seems not to react to catch up with the new data type change within databases. Some of them cannot retrieve XMLType column data correctly from XML-enabled database; some of them cannot retrieve all column data and shown on the pages because of new XMLType; some of them even cannot generate the web application because of the unsupported data type. They become bottlenecks for those development tools. It requires web application developers to find out corresponding workarounds to deal with XMLType data processing. The key

objective of this research paper is to explore different potential types of technologies for XMLType processing and investigate into workable implementations with those web-based application development tools. It is expected to solve the unsupported new XMLType data type issues by means of some workarounds for more efficient web based application development on XML-enabled relational databases.

2.5 Research Methodology

In order to address the research problem and investigate possible solutions for achieving the objective, we will use different research methodologies.

Theory – It is necessary and compulsory to give out all definitions of relevant terminologies including data-centric data model, document-centric data model, native XML databases, XML-enabled databases, declarative query languages, application programming interfaces (APIs), PL/SQL and all definitions for possible solutions of the research objective. They serve as the basics for further discussion in the research paper. Some of them have already mentioned and defined in above sections and the others will be stated afterwards.

Comparison – Since the main purpose of the research is to investigate different types of web application development tools, it is essential to have comparison method. We need both comparisons before new XMLType data introduction and after new data type introduction. A general comparison on several criteria among three different types of web development tools will be implemented in order to investigate features and abilities of them. It is obvious that these web application development tools developing teams and lots of XML communities have already released a variety of proposals on either workarounds or approaches for XMLType data processing (both extraction and manipulation) for web application development. That makes comparison method be an essential procedure. Besides the general comparison, further comparison on these different workarounds or approaches becomes the key aspect for this research report. Only if compare different approaches can we find out their advantages and disadvantages in order that a better choice can be made based on the result from those comparison.

Generalization/Specialization – Sometimes it is difficult to get the correct solution for new requirements immediately. Before we look into how to achieve the requirement of XMLType data handling within web development tools, we had better first explore all possible approaches when we are dealing with XML data in different situations. It may make it simpler if we can solve the XML data handling problems before we consider new XMLType data type integration within existing web application development tools. After we fully understand all features of XML data storage and processing, it may be better to consider how to fulfil the new data type requirement based on the previous acknowledgements. That is the generalization and specialization method.

Examples – Only presenting the theories of workarounds or approaches is not efficient to thoroughly make sense to some extent. It is also necessary to implement these possible solutions into real web application example. They can also be applied when we do comparison between different approaches either for development with or without XMLType data. For basic comparison among tools uses ordinary back-end relational database without XML data as data source for web application development while key comparison on solutions and approaches uses database table with XMLType column as example data source. Those examples would be essential blocks to build up our discussion. We supposed to concentrate on these example application developments and observe their performances so that we can get a conclusion on sufficient XMLType handling with different web-based application development tools.

3 A Brief Outline of Three Popular Web-based Application Development Frameworks

As we all seen above, we have a variety of choices for web-based application development tools. The three types of development tools we mentioned are some typical examples in the industry. They all provide developers necessary functionalities and tools for easy-to-use development environment. However, they also have their own unique features. In this section, we will briefly compare and evaluate them based on different criteria.

3.1 Installation

Easy installation and configuration is the first key aspect for end-users and developers. For those lower-level end-users or fresh users of the tool, this could be a big challenge for them to start using the tool.

3.1.1 Installing Oracle Application Express (APEX)

Among these three tools, Oracle Application Express is the easiest one to setup because of its integrative feature. Certain entry level Oracle database such as Oracle Database 10g Express Edition has Application Express already integrated into it. Once Oracle Database EX has been installed, the web-based application development tool (APEX) is already installed and configured and ready for use. However, if there is a need to install and configure Oracle Application Express into an existing Oracle Database end-users must separately install Oracle HTTP Server and Oracle Application Express. Although Oracle offers end-users companion CD which includes installation package, they still have to modify PATH environment variable and configure HTTP server settings based on Oracle database to run APEX. That may relatively cost more extra time for entry-level web-based application developers.

3.1.2 Installing WaveMaker (formerly ActiveGrid)

Both ActiveGrid and WaveMaker, on the other hand, can be easily installed by using software installation package. There is no need for end-users to post-configure the development environment after installation. Even if developers want to change the HTTP server settings, they can easily modify the server configuration under corresponding software menu item. Overall, this type of web-based application development tool is especially designed for those end-users who are looking for an efficient and flexible development solution.

3.1.3 Installing Ruby on Rails

Lastly, the development framework on programming language (Ruby on Rails) is

somehow “manual” to setup. Programming language Ruby and web framework Rails is separately installed and configured. If end-users don't have the patience to get Ruby on Rails running manually, they can also try one of the pre-packaged solutions called Instant Rails. It includes everything in one bundle: web server, database, Ruby, and Rails.

3.2 Usability

Easy-to-use is another key factor that dramatically influences the comfort and efficiency of the web-based application development processes. All levels of web application developers want to find a user-friendly and comfortable development environment in order to increase the efficiency and have an enjoyable developing experience. Its easy to use browser based application builder enables developers and non-programmers to develop and deploy data driven web applications in very little time.

3.2.1 Using Oracle Application Express (APEX)

Oracle APEX is relatively designed for entry-level end-users so that most of the development processes can be realized by using pre-defined wizards. For instance, a simple application can be simply created through “Create Application” wizard under “Application Builder” menu item. Seven steps are needed to fulfil the whole process of application generation including name, pages, tabs, shared components, attributes, user interface and confirmation. The application editor panel provides user-friendly interface where essential functionalities and indicators are located.

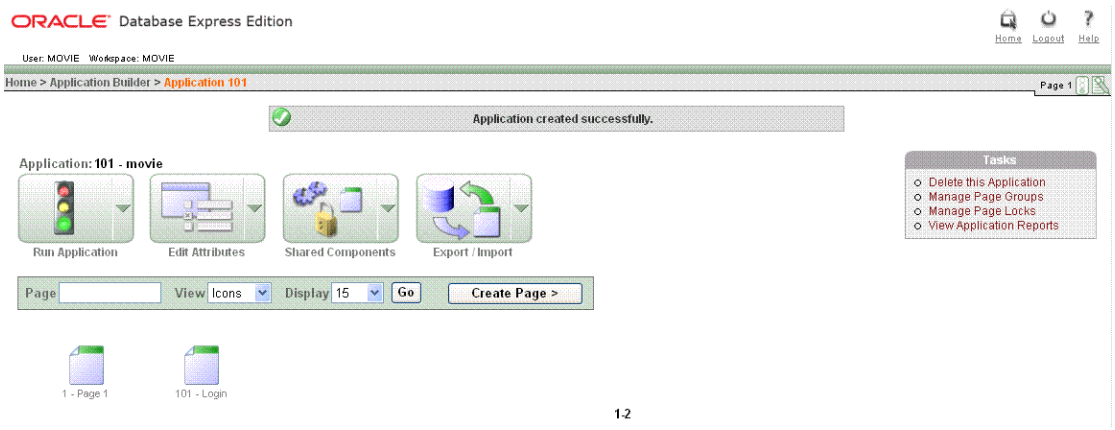


Figure 17: Screenshot of Oracle APEX application builder

Oracle APEX uses pre-defined methodology to render and process pages across the entire application, which speeds up the assembly of pages. Single page structure and definitions can be also declared and modified through “Page Definition” section.

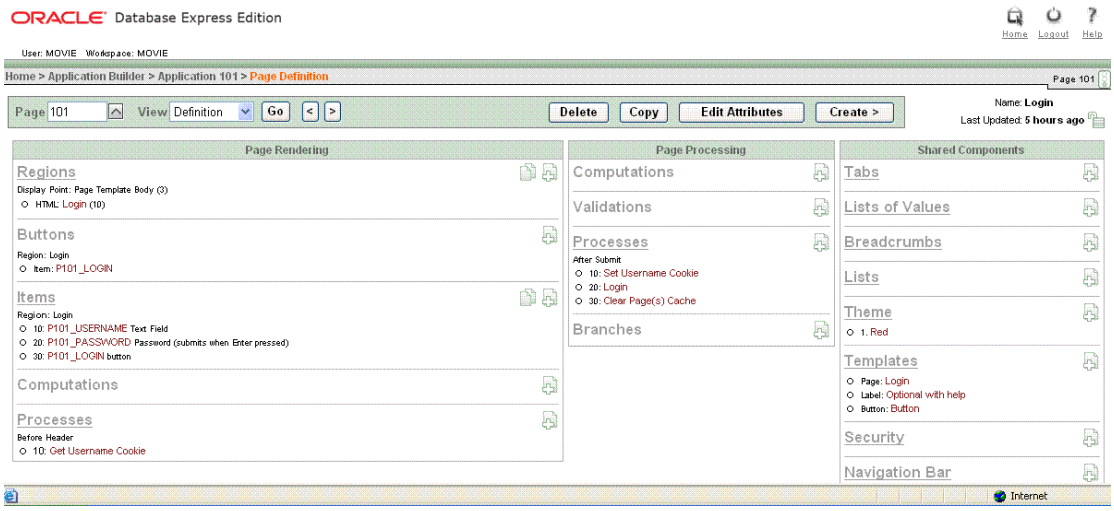


Figure 18: Oracle APEX page definition section

The way Oracle APEX uses is separation of user interface, application logic and data access. It uses templates for pages, regions, reports, labels and other elements of an application and encourages developers to separate the look of application from logics and functionalities. It makes maintenance easier since understanding how a page works does not require looking through code or scripts. In Oracle APEX, a developer can quickly tell what function a page performs and how it works, even if he's never seen the application before. Developers can also work on the logic of the application before they decide on the user interface and the look of the application can be changed only by changing the templates rather than every page or component in the application because the separation of different components.

3.2.2 Using WaveMaker (formerly ActiveGrid)

ActiveGird and WaveMaker are somehow similar with Oracle APEX to some extent. As shown in Figure 8 within Dashboard tab of WaveMaker, it also divided the application development into three separate parts which are "Pages", "Services" and "Data models".

Figure 19: Dashboard editor tab of WaveMaker

Developers can add, modify and delete building blocks of application (pages), specifications of all operations (services) and definition of the connection to each database (data models) by only using the user-friendly functions within dashboard editor section. Similar with Oracle APEX, WaveMaker also provides “Design” editor for page definition as shown in Figure 4 where drag-and-drop feature is implemented for easy page layout definition and page elements allocation. Page element property and page model give developers an object-oriented application development environment, even though WaveMaker is intended for developing web-based applications.

3.2.3 Using Ruby on Rails

Ruby on Rails, as an open-source web-based application development framework, is different from those integrated development environments above. It is more or less a conceptual structure which illustrates a method to develop web-based application. It provides developers essential components such as HTTP server, database connection and programming language support. Instant Rails is the typical example which we investigate in this paper. The central control panel look like as follow:

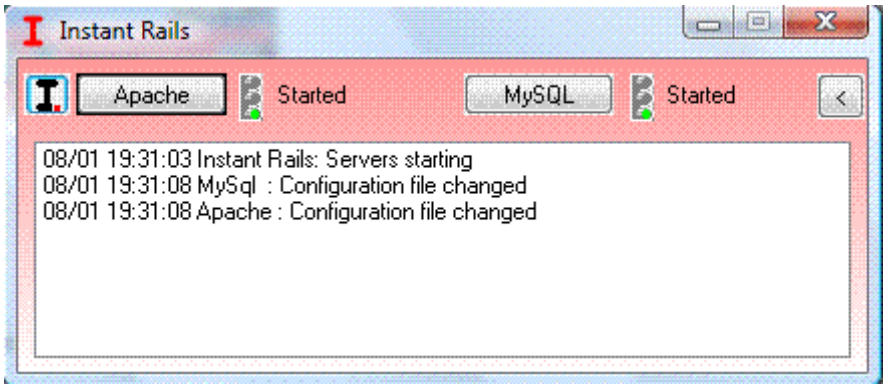


Figure 20: Instant Rails control panel

The panel gives clear indicators for HTTP server and database status. All necessary developing methodologies and functions are available under “I” menu. Even if developers use one-stop Ruby on Rails runtime solution, they still have to use Rails command line to generate, modify or delete web-based applications and Ruby command to deal with database connection or application modifications. That makes it hard to get started quickly for entry-level developers or end-users such as staffs. Furthermore, there is no page designer interface or data model monitor. Since Ruby on Rails use MVC (Model-View-Controller) structure, ruby scripting files and rhtml web page files are created under corresponding folder (models, views and controllers folders) throughout the whole application development process. Developers have to manually use text editors to modify either web pages code, function code or controller code. That means developers have to look into and know back-end code and scripts, which is also not suitable for entry-level developers.

3.3 Performance

Performance is also a key factor which can greatly influence user’s choice especially because the duration of development circle is related to the cost of application development. In this case, developers should consider both developing and deployment period durations. In order for fair comparison of three different types of development tools, we make a small project task to observe which development tool can finish it better. The project task that we’re going to build in this case uses Oracle 10g Express Edition as data source for all three development tools. This simple task begins with a single search/list page. We have a database that contains movie related information. We want to create a simple page that allows users to search for a movie by certain key words; display a list of movies that match the search characters, add new movie, delete existing movie and modify certain movie information. Later we’ll add more functionality to this basic application.

3.3.1 Rapid Developing with Oracle Application Express (APEX)

As we expected for Oracle APEX, developers only need to follow new application generation wizard steps and choose what contents (pages, components, etc.) within the initial layout. In this case, developers only need to add report and form pages to achieve the requirements. A basic application with CRUD (create-read-update-delete) and even search functions is automatically created. The whole developing process only takes 15 minutes.

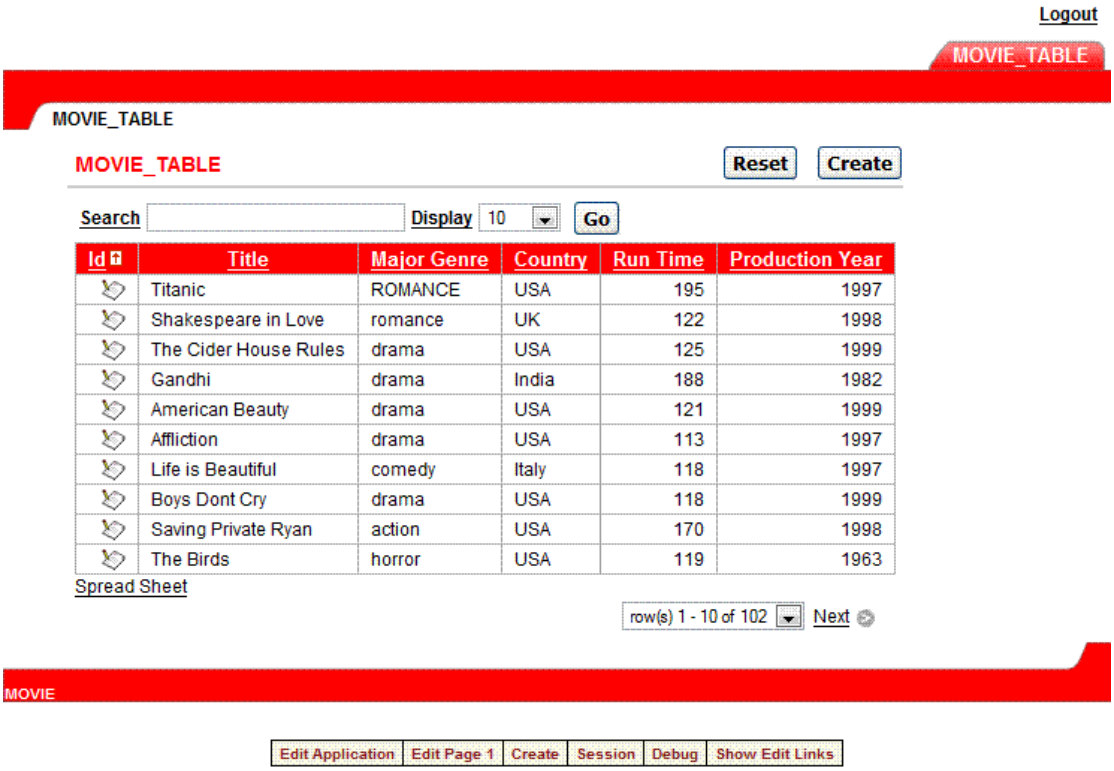


Figure 21: Movie report page

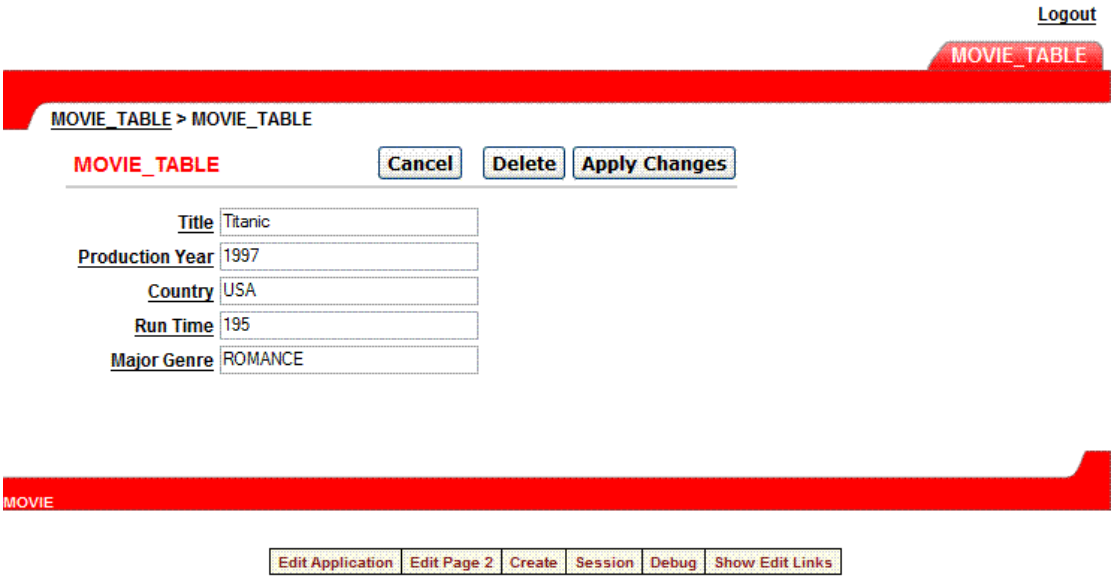


Figure 22: Movie form page

The modification after developers create the application can be realized through page definition (Figure 9). For instance, developers only need to modify column attribute to change “modify” icon to “Title” text link within Movie report page definition. They can also make default field(s) (e.g. Run Time) to be hidden within Movie form page definition. The final application looks as follows:

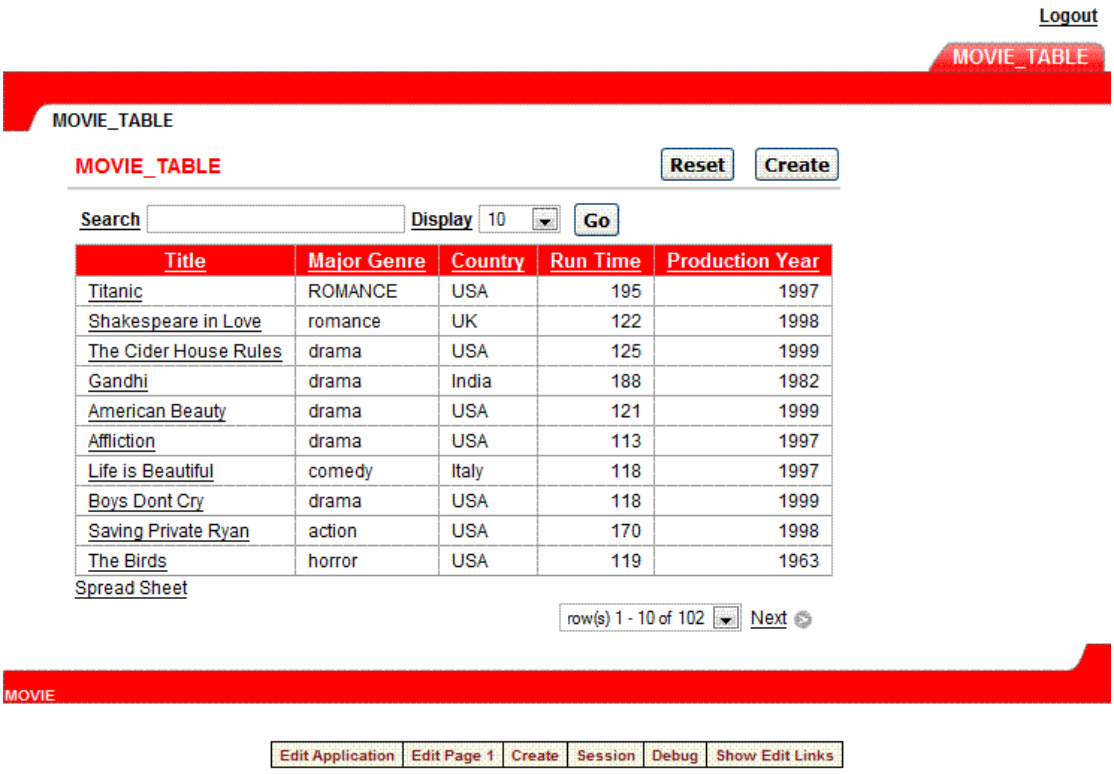


Figure 23: Modified Movie report page

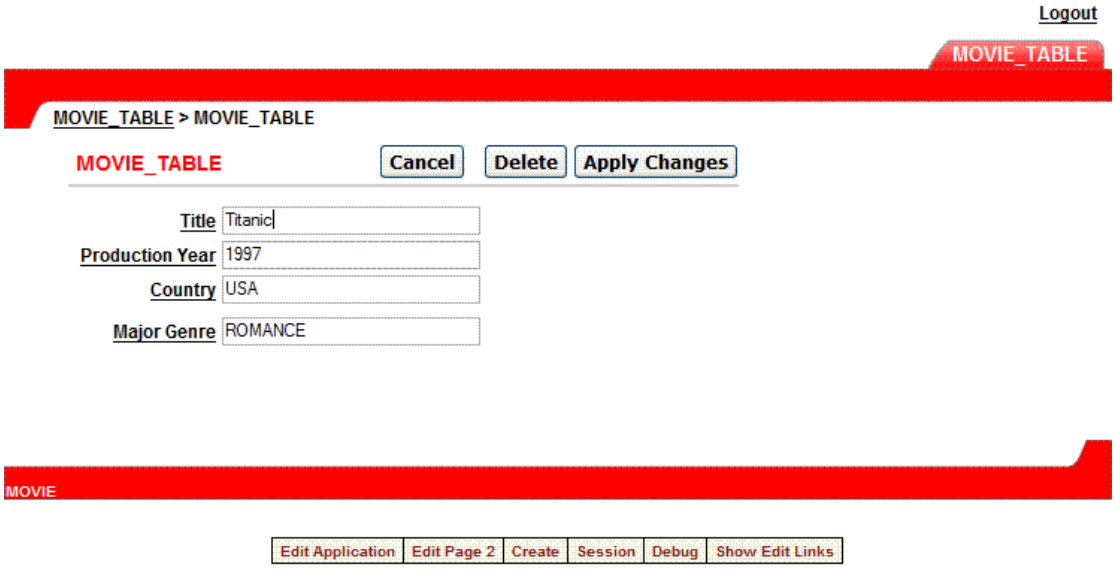


Figure 24: Modified Movie form page

3.3.2 Rapid Developing with WaveMaker (formerly ActiveGrid)

The second tester, WaveMaker, looks much more sophisticated to use than Oracle APEX. Unlike Oracle APEX which directly reuse data model from its own database, developers have to import the data model from external databases (advantage to some extent) and created service calls. Due to the newly released development tool after ActiveGrid, it hasn't already integrated some useful automated features in ActiveGrid.

For instance, developers only need to import data model from database and ActiveGrid 2.x automatically generates web pages with default basic functions provided (list, create, edit, delete, search) which are exactly what entry-level developers need. However, new WaveMaker 3.0 only generates basic "services" for imported data model, but not the actual pages. According to Geremy Cohen, the Community Development Manager of WaveMaker there are “plans for creating autogenerated CRUD pages in a future release” *(Cohen, 2008)*. Although developers have to create web pages manually, user-friendly drag-and-drop Widget Palette feature makes layout designing pretty easy and rewarding according to the developers’ requirements. It gives web-based application developers a feeling that they are using a professional object-oriented development environment. Basic database-related service calls are already available for use after developers import the data model. For small applications a first running version can be produced in no more than 30 minutes.

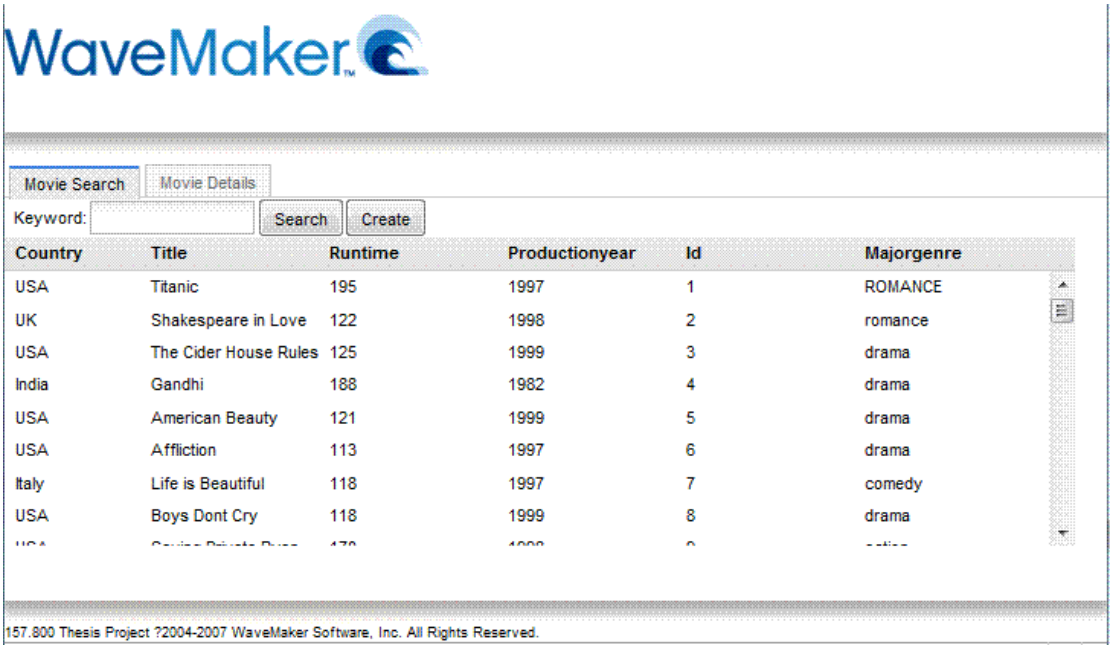


Figure 25: Movie search page by WaveMaker

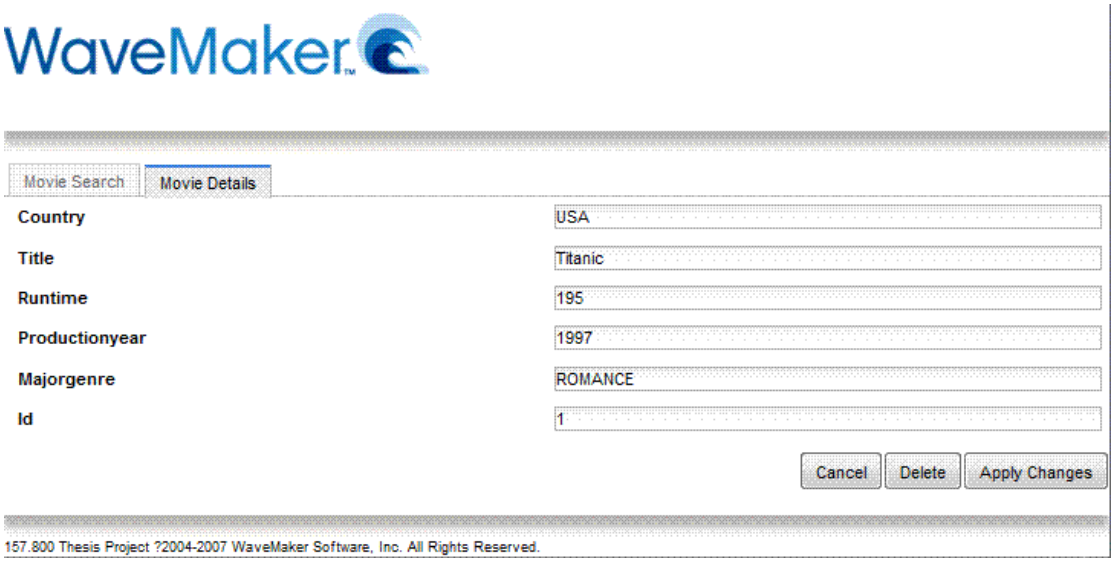


Figure 26: Movie details page by WaveMaker

Compared with Oracle APEX application modification, WaveMaker application is somehow easier. During the development processes, developers have already familiar with existing page elements, service calls and operations. They can quickly find the item or place which or where they need to modify according to the requirements. In this case, we easily get rid of "Id" column from the movie list and re-order the sequence of table columns which only needs to modify the "dataFields" property belonging to the "movieList" item on the MovieSearch tab to be "title, majorGenre, country, runTime, productionYear". As we did in Oracle APEX, we also remove the runtime attribute and re-order the sequence of form elements from the movie details form only by changing the "dataFields" of form property to be "title, productionYear, country, majorGenre". The final application looks as follows:

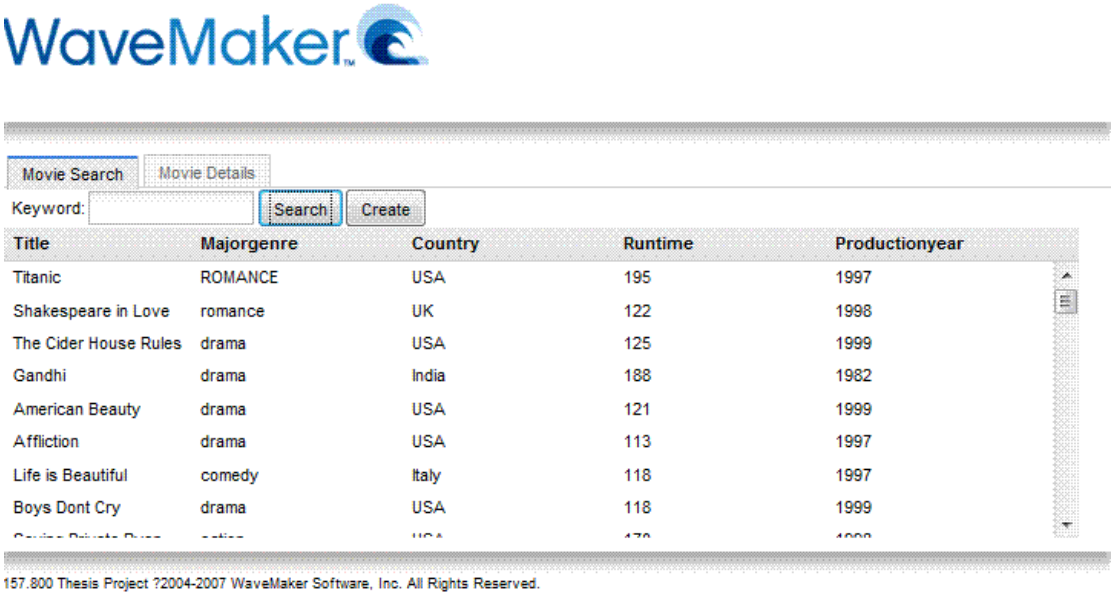


Figure 27: Modified movie list page on WaveMaker

Figure 28: Modified movie details page on WaveMaker

3.3.3 Rapid Developing with Ruby on Rails

The last development tool to test is Ruby on Rails. The natural feature of this type of web application tool has already determined the difficulty for developing with this tool. Before the developers start implementing application, they have to confirm all necessary framework components. Although some “lazy” Ruby on Rails package such as instant Rails can provide an easy-to-use basic framework which already includes almost all necessary components such as Ruby, RubyGems, Rails, and MySQL/SQLite as default, they still have to consider more exceptions. In this case, since we use Oracle 10g Express Edition as our testing database, Rails Oracle Library is necessary to allow your Rails applications to talk to your Oracle database server. Furthermore, an ActiveRecord-oracle-adapter is also compulsory as database adapter. Each application database settings must be manually configured in a specific file. A good news is that Rails framework also provides some useful scripts (script/generate scaffold) to generate basic web application with CRUD functions. Scaffold generation script automatically create basic model, controllers, and view files according to the MVC model for the part of the application that use the corresponding table specified in ruby command. The whole development process takes about 40 minutes.

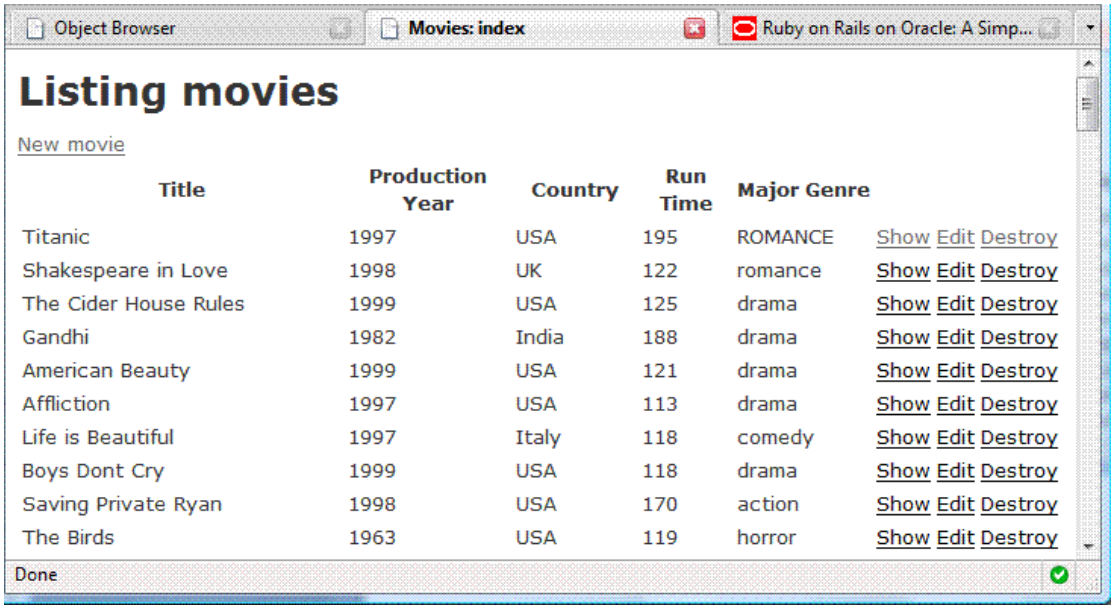


Figure 29: Movie list page by Ruby on Rails

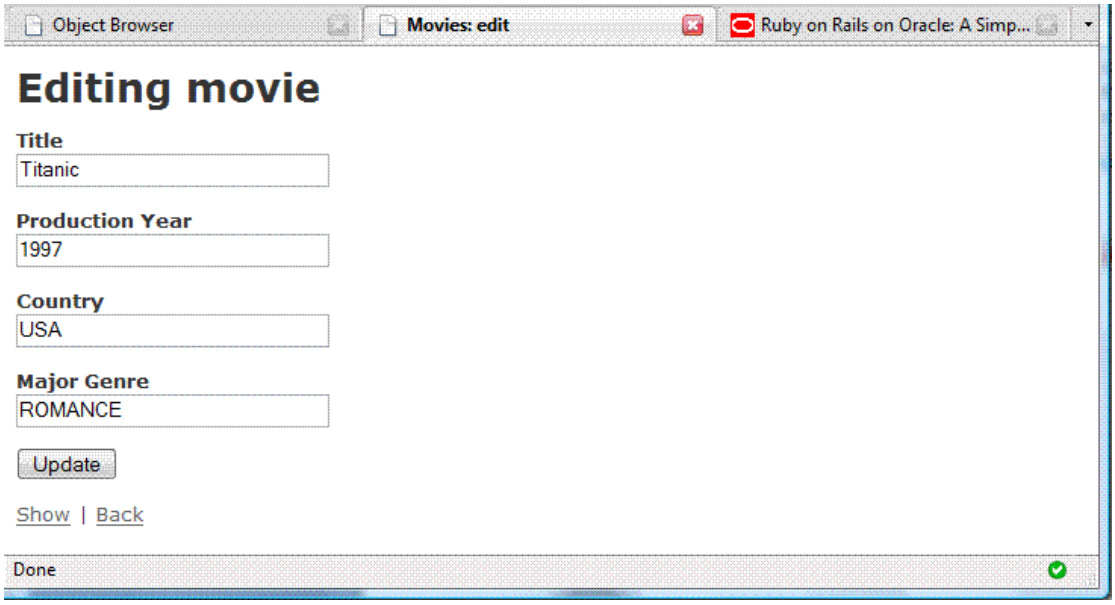


Figure 30: Movie edit page by Ruby on Rails

Unlike those two development tools above, Ruby on Rails framework does not provide any interface where developers can see virtual structure of pages or element properties belonging to each page. All they can do is manually modify code within corresponding files. In this example, we add a “search” function onto the existing application. The layout code is changed in the ruby html file (e.g. index.html.erb/index.rhtml) under views folder where we can reuse XHTML code to generate the search form. Search function for passing keyword value from search form to find corresponding records from database and return movie object instance is located in the ruby controller file (e.g. movie_controller.rb). Developers also have to manually define a search method to implement this function in the controller file.

Finally, the search results view can use the same code as in the ruby html file as a new index page. In this case, there is no need to modify model part because only one data model is added into this simple prototype application. The basic application modification processes on Ruby on Rails focuses on these steps. It gives developers clear back-end application structure and developing flexibility but also requires developers have more knowledge on programming language (Ruby) and development tool structure (Rails). The final application looks as follows:

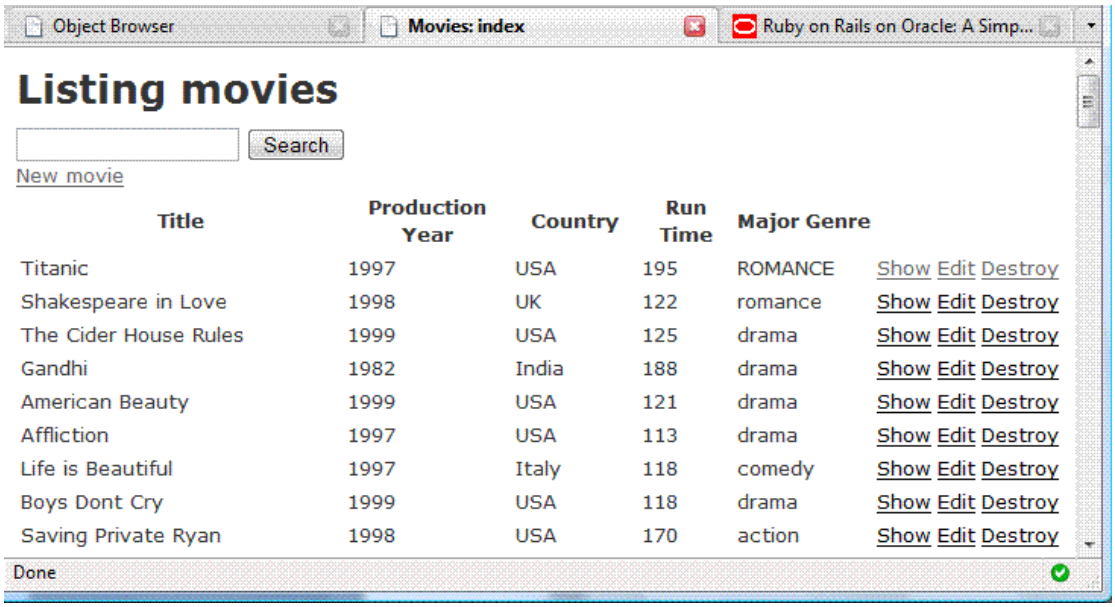


Figure 31: Final movie index page by Ruby on Rails

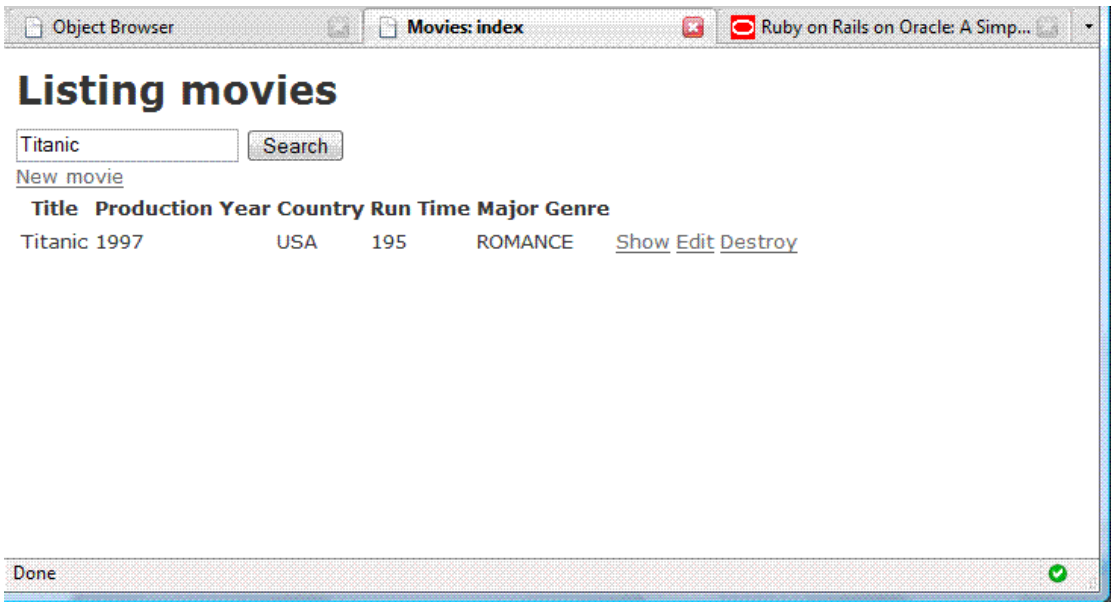


Figure 32: Search Result Page on Ruby on Rails

3.4 Deployment

The deployment methodology of a web-based application development tool also plays an important role in development tool decision. Generally speaking, all application development tools should have provided certain deployment method which is able to help developers easily deploy the finished application so that end-users can begin using it.

3.4.1 Rapid Deployment with Oracle Application Express (APEX)

Oracle APEX still continue taking advantage of its integration features and shows us how easy for developers to deploy application built on it based on Oracle database. The whole process only includes the following steps *(Oracle, 2005)*:

- Collect your supporting object definitions (CREATE object statements, sample data, images, files, and so on),
- Define your supporting objects (including the object definitions you collected in the previous step as well as other supporting objects, such as messages and prerequisites),
- Export the application with its supporting objects,
- Import and install the exported files.

Exporting and importing processes are fairly easy. For exporting data, developers can simply select Export DDL. It is saved to a file, zip file, a SQL Developer Worksheet, or the Clipboard. To Import, developers can select Import Application function. A wizard will guide you through selecting your file and you will be given the opportunity to rename the application, set a new Application Alias, Build Status and Application ID. Using Application Express greatly simplifies the deployment process. It enables developers to migrate the application definition, associated files, and all supporting objects in those steps above. The rest before end-users can use the application is to create end user account(s) and publish the application URL. Besides, all those processes can be done within Oracle Application Express.

3.4.2 Rapid Deployment with WaveMaker (formerly ActiveGrid)

WaveMaker Visual Assembly Studio is especially designed for development and testing and includes an evaluation copy of the Rapid Deployment Framework which can also greatly implement application deployment task. As WaveMaker announced, WaveMaker Rapid Deployment Framework is a standards-based framework that deploys Java applications which central IT can maintain and extend with their existing skills and tools, see *(WaveMaker, 2007)*. One of the key features of WaveMaker Studio, one-touch deployment, exactly shows the deployment method works. It uses real-time deployment to local Tomcat servers for testing and staging. WaveMaker applications can be deployed on a .WAR file and in a local, on-site or on-demand deployment. The studio allows developers to export projects as Java WAR files (also

called the *WaveMaker Deployment Framework*) that can run in any Java container. This WaveMaker WAR file is simply a "packaged" copy of the web application root directory of a WaveMaker project. To create a WAR file from a WaveMaker project, developers only need to execute “Generate WAR file” methodology within administration panel in Dashboard tab as shown in Figure 8. The WAR file will be named <project>/dist/<project>.war. Later on, we can use the deployment tools provided by your application server to deploy the WAR file to any Java server, including Tomcat, Weblogic, Websphere and JBoss. It gives the application more flexibilities and compatibilities.

3.4.3 Rapid Deployment with Ruby on Rails

Similarly, Ruby on Rails provides lots of deployment choices. Due to the common application development framework feature of Ruby on Rails, there is no best deployment strategy in the industry. But, certain “lazy” Ruby on Rails package as Instant Rails has already shown us some popular or reasonable rails application deployment environment that is Apache-Mongrel combo. This application server combination is exactly used for local deployment when developers need to test the unfinished web applications. It shows the deployment of a single instance of Apache running on a Windows server that proxies requests to various instances of Mongrel running as Windows services on higher ports *(Hogan, 2006)*. Besides installation of both Apache and Mongrel, we also have to manually configure Apache server to proxy Rails applications. This may gives some entry-level developers some challenges when they want to setup the environment for deployment. A good thing for Ruby on Rails application is that developers can reuse the project structure and directly put them into the live application environment which is setup using the deployment strategy mentioned before. The whole process highly depends on the level of developers.

3.5 Some Final Observations

Overall, although those three types of web application development tools have different features, different development methods and different deployment strategies, we can still notice some common points and certain rules to make our decisions on.

3.5.1 Oracle Application Express (APEX)

As we can notice, Oracle APEX has its inborn advantages which make it look very easy to build web applications based on Oracle databases. Developers can use those pre-defined wizards and integrated methodologies to create and deploy a web application which has all capabilities they need. There is no need for end-users to look into back-end code or know how it works. That makes it relatively ideal for both power users (developers) and non-experts (staff). It provides both two categories of users an easy-to-use tool for efficient development and deployment. This includes the flexible integration of Oracle SQL and PL/SQL for implementing applications with more complex functionality without the need to use any further programming

language. Certain unsatisfied factor for Oracle APEX is also obvious. On the other hand, the close coupling of the embedded development environment with the Oracle DBMS reveals the major limitation of this approach. It is only suitable for developers who want to create applications that a centred around an Oracle database.

3.5.2 WaveMaker (formerly ActiveGrid)

WaveMaker as a specially-designed web application development tool for Web 2.0, also provides user-friendly integrated development environment. It gives developers a professional-like OO programming interface which includes several fancy features such as drag & drop assembly and live layout. Though it does not offer a quick pre-defined page layout as Oracle APEX does, such humanized features also gives developers easy-to-use facilities. It is somehow more flexible and rewarding for designing a web application according to their preferences. On the other hand, WaveMaker supports almost all popular databases such as MySQL, Oracle, DB2, and SQL Server. The separation of page design, data model and service calls is also an advantage for better understanding of application structure. It is a standard application development tool for all levels of developers. One might regret that WaveMaker now focuses on Java and JavaScript as programming languages for customizing and implementing additional functionality. Its former version, ActiveGrid, still supported three languages (Python, PHP, and Java). That decreases the flexibility of WaveMaker.

3.5.3 Ruby on Rails

At last, Ruby on Rails framework does not make a good first impression when developers try to install all necessary components and configure the environment. Fortunately, certain package such as Instant Rails prepares a ready-to-go developing environment for new users. Similar with WaveMaker, it supports popular database connection with Rails application but that also requires developers installing corresponding database adapter to make it work. The whole developing process relatively depends on the level of developers since they have to manually modifying xhtml or ruby code to meet application requirements. It gives developers more flexibilities but it is more likely suitable for power users (developers).

4 Towards XML Data Processing

Nowadays, XML has already been propelling the transformation of World Wide Web into the next generation. The key strength of XML is its ability to represent both structured data and unstructured data. As we mentioned, XML data has gradually become a popular interchanging information media and widely used as one of important data type within variety of applications. Due to the high demanding of this new data type storage and implementation, lots of popular databases have already introduced new XMLType and provide necessary mechanisms to handle XML data/document. The vision of those XML-enabled databases is to unify the two traditionally separate worlds of data and content management by building on this strength of XML data. DB2, Oracle and SQL Server are typical databases that support XML data by a special data type.

All three web-based application development tools that participated in the comparison (Oracle APEX, ActiveGrid/WaveMaker, Ruby on Rails) support the Oracle DBMS as a backend database tool. In the practical research for this thesis therefore I focussed on Oracle's XMLType as a typical example that can give more insight on how web-based application development tools can handle XML data. For the sake of comparison, I used an identical database with data of type XMLType. This database was implemented using the Database 10g Express Edition of Oracle's DBMS.

4.1 Oracle XML DB

Oracle XML DB *(Oracle XML DB, 2003)* has been first introduced with the release of Oracle Database 9i. It provides the Oracle DBMS new capabilities for the efficient storage, retrieval, querying, generation, and management of massive volumes of XML data. Oracle XML DB has also provided deep integration of XML Schema, XMLType, both structured and unstructured storage, content repository, PL/SQL and management capabilities. With the new version Oracle database releases, XML DB continuously evolve with more versatile, scalable, and efficient features in different areas. In the new Oracle Database 10g Release 2, Oracle XML DB introduces an efficient implementation of standards-based XQuery capabilities, a versatile schema-based resource metadata facility, a set of new SQL functions for DML operations on XML data, and more *(Oracle, 2005a)*. The Oracle XML DB storage has two primary pillars which are XMLType tables or views and Oracle XML DB repository (file, folder or URL metaphor) as shown in the figure blow.

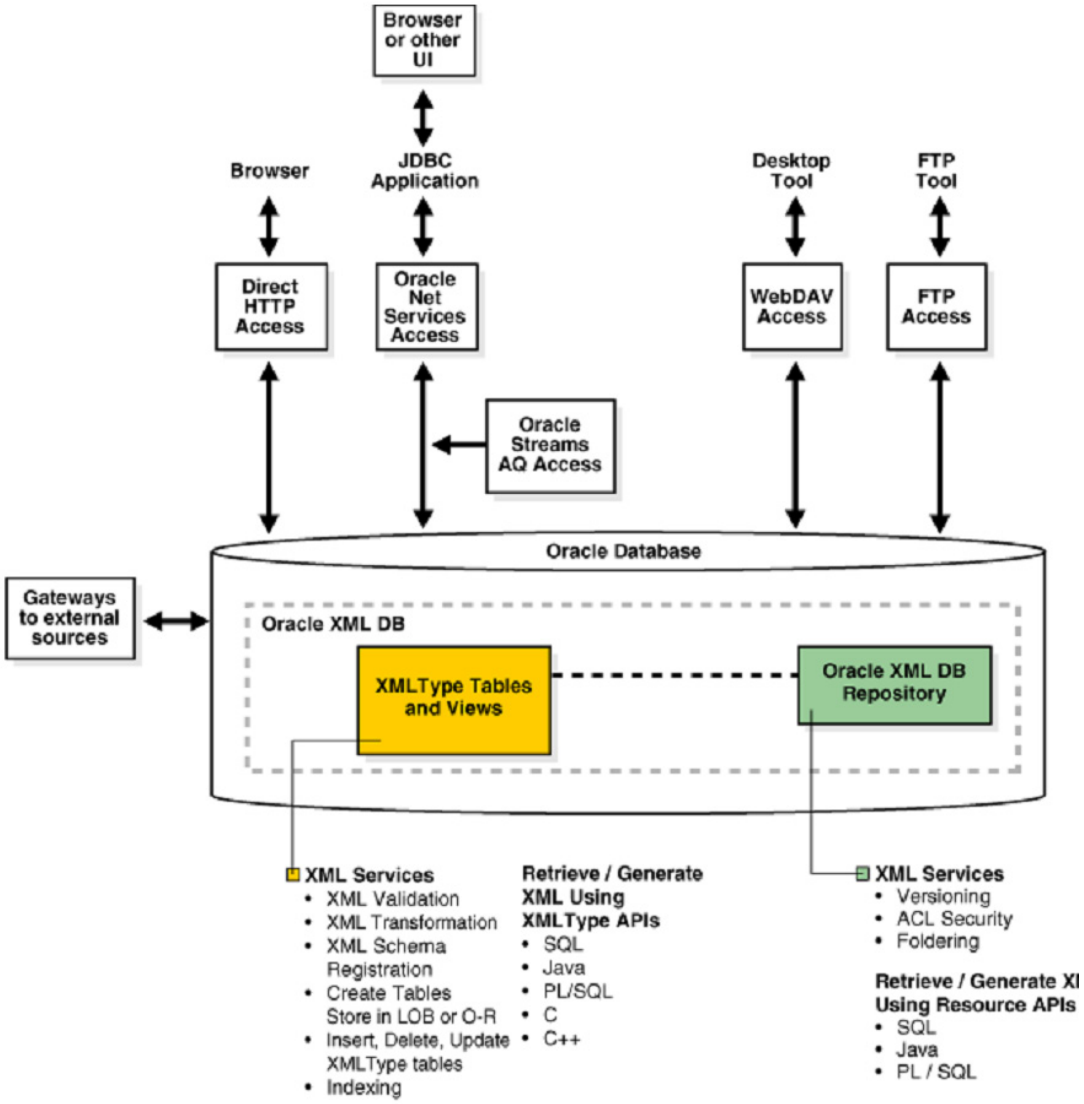


Figure 33: XMLType Storage and Repository

As we can see from this diagram, a variety of functionalities are also delivered together with the new data type introduction. On the higher layer, native database support for industry-standard, content-oriented, protocols including FTP, HTTP and WebDAV makes it possible for moving XML content into or out of the Oracle Database. On the lower layer, several industry-standard APIs that allow programmatic generate, retrieve and manipulate of XML content from SQL, Java, PL/SQL and C. Recent release of Oracle Database even support native XML query language – Xquery and more new SQL functions on DML operations on XML data. All these abilities bring the enterprise class management capabilities of the Oracle database including reliability, availability, scalability and unbreakable security to bear on XML content.

For the research implementation here, I used Oracle Database 10g Express Edition for the back-end storage of XMLType data. External XML documents containing sample data (from an Accounting application) are imported by using the pre-defined functionality provided by the XDB_Utilities Package of Oracle's XML DB and saved

into an XMLType column within a simple table, called xmldemotable. The routines in the XDB_Utilities package can be classified as follows according to the functionality they provide *(Oracle XML DB, 2003)*:

- Subprograms for getting the content of an OS file
- Subprograms based on Oracle XML DB Resource APIs
- Subprograms for DOM Manipulation based on PS/SQL DOM API

In this case, we implement the first type and the whole process follows the usage of getXMLFromFile() subprogram within XML DB Utilities Package to implement PL/SQL script. This script demonstrates how first register an XML schema using the subprogram and then use it to load multiple instances of XML documents (with sample data from an Accounting application) into the database. The XML data is finally stored into the "xmldemotable" table which can be queried to view the XML content for web application development. The final table with XMLType column is illustrated as follows:

Column Name	Data Type	Nullable	Default	Primary Key
ID	NUMBER(4,0)	No	-	1
NAME	VARCHAR2(30)	Yes	-	-
CONTENT	XMLTYPE	Yes	-	-
				1 - 3

ID	NAME	CONTENT
1	acct1.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>0</ACC_NO> <CST_ID>0</CST_ID> <ACC_BALANCE>1040</ACC_BALANCE> <ACC_CREDITLIMIT>3.14159</ACC_CREDITLIMIT> <ACC_CREATEDATE>1967-08-13</ACC_CREATEDATE> <ACC_CARDTYPE>MASTER</ACC_CARDTYPE> <ACC_ENABLED>true</ACC_ENABLED> </ACCOUNT>
2	acct2.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>7</ACC_NO> <CST_ID>0</CST_ID> <ACC_BALANCE>1040</ACC_BALANCE> <ACC_CREDITLIMIT>34.14159</ACC_CREDITLIMIT> <ACC_CREATEDATE>1977-08-13</ACC_CREATEDATE> <ACC_CARDTYPE>VISA</ACC_CARDTYPE> <ACC_ENABLED>true</ACC_ENABLED> </ACCOUNT>
3	acct3.xml	<ACCOUNT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://otn.oracle.com/account.xsd"> <ACC_NO>8</ACC_NO> <CST_ID>8</CST_ID> <ACC_BALANCE>888</ACC_BALANCE> <ACC_CREDITLIMIT>8.88888</ACC_CREDITLIMIT> <ACC_CREATEDATE>8888-88-88</ACC_CREATEDATE> <ACC_CARDTYPE>test</ACC_CARDTYPE> <ACC_ENABLED>test</ACC_ENABLED> </ACCOUNT>

Figure 34: Sample table with XMLType column in Oracle

4.2 APIs for XML Data Processing

Application Programming Interface (API) is an important issue when a developers deals with back-end data source for applications. They are originally proposed by W3C in order to efficiently support the rapid growth of the web and e-commerce. SAX (Simple API for XML) and DOM (Document Object Model) are the most popular APIs for processing XML data. While SAX parses through the XML document like a data stream, DOM builds an in-memory representation of a parsed

XML document in form of an abstract tree structure (known as the DOM model of an XML document), see *(Chen and Tompa, 2003)*.

Most of current XML tools work with both APIs (SAX and DOM) for XML document processing. Those two application programming interfaces also serve as models of how XML data is parsed. Parsing an XML document can be very resource-intensive. Therefore, the overall performance of XML operations is greatly influenced by the performance of the XML parser. SAX and DOM, as two typical models, should consider many issues regarding XML converting into relational format or XPath expression evaluation. Due to the infrastructure feature of parsing issue and major intention of this paper, the performance issue on XML parsing may not be concerned to some extent. More effort will be made on the XML data processing in a higher layer. The basic theory for XML APIs is mapping traditional databases into data-centric XML documents because of its regular data storage structure. As stated by *Laddad (2000)*, "with an XML API for databases, we can make the database look like an XML document; these APIs present the database as a virtual XML document. We are at the most basic concept of object-oriented design: it is the interface - not the implementation - that matters. In our situation, the tools using such an XML API need not care whether they are operating on a database table or an XML file". Based on the two XML APIs, several proposals have come up for facilitating XML processing with various languages.

4.3 Enhancing PL/SQL for XMLType Processing

PL/SQL stands for "Procedural Language extensions to the Structured Query Language". SQL has already become a ubiquitous structured query language for both querying and updating of relational database. Oracle Corporation introduced PL/SQL in order to overcome some limitations from SQL and to provide a more powerful solution for those developers who are attempted to build mission-critical applications based on the Oracle database. Oracle's PL/SQL language has several defining characteristics, see *(Feuerstein, 2002)*:

- a highly structured, readable, and accessible language,
- a standard and portable language for developing with Oracle,
- an embedded language, and
- a high-performance, highly integrated database language.

With these features, PL/SQL is already at the heart of much of Oracle's software and serves as the programming language within the Oracle Developer toolset. It is tightly integrated into SQL which allows developers execute SQL statements directly with procedural program without relying on intermediate API (Application Programming Interface) such as JDBC or ODBC. Since PL/SQL figures prominently as an embedded technology in almost every new release by Oracle Corporation, application developers use PL/SQL to perform a variety of programming functions including

implementing business logics in the Oracle server with PL/SQL-based stored procedures and triggers, implementing and automating database administration tasks, linking web pages to Oracle database and generating/managing XML documents entirely within the database. It supports a full range of datatypes from number to string and including complex data structure such as records, collection and XMLType as well.

To deal with XMLType data with PL/SQL, specific XMLType APIs are provided within Oracle Databse for XMLType for developing applications that run on the server. They are PL/SQL DOM API for accessing XMLType objects, PL/SQL XML Parser API for accessing the contents and structure of XML documents and PL/SQL XSLT Processor for transforming XML documents to other formats using XSLT. These PL/SQL APIs in Oracle XML DB provide native XML support within the database. Since XMLType is a system-defined opaque type for handling XML data within Oracle database, it has predefined member functions and procedures to extract XML nodes and fragments. With the help of these predefined functions and procedures, developers can create XMLType tables, columns and views; insert XML documents into as we did for creating out “xmldemotable”; access and manipulate XML document/fragment from XMLType column. A simple example with PL/SQL scripts looks like as follow,

```
CREATE TABLE Xml_tab ( xmlval xmltype);

INSERT INTO Xml_tab VALUES (
    xmltype('<?xml version="1.0"?>
                    <EMP>
                        <EMPNO>221</EMPNO>
                        <ENAME>John</ENAME>
                    </EMP>'));

INSERT INTO Xml_tab VALUES (
    xmltype('<?xml version="1.0"?>
                    <PO>
                        <PONO>331</PONO>
                        <PONAME>PO_1</PONAME>
                    </PO>'));

-- now extract the numerical values for the employee numbers

SELECT e.xmlval.extract('//EMPNO/text()').getNumVal() as empno
    FROM Xml_tab
    WHERE e.xmlval.existsnode('/EMP/EMPNO')    = 1;
```

A list of predefined functions and procedures for XMLType data processing is provided within Oracle database. They help application developers on dealing with

XMLType data stored in Oracle database. It is a potential solution for web application development tools to retrieve XMLType column data and manipulate XMLType data for certain functionalities.

4.4 Incorporating XQuery for XMLType Processing

The W3C community spent much time and effort on developing XQuery *(XQuery, 2007)* as a standard for querying XML data. Following a full call for proposals a query language was designed that meets a number of well-specified criteria. XQuery makes use of XPath *(XPath, 2007)* to navigate in XML dcouments. It uses many features that have been proposed by a range of researchers in earlier proposals for a XML query language. For a detailed survey, see for example *(Abiteboul, Buneman, Suciu, 1999)* and *(Chamberlin, 2002)*. Since the release of W3C recommendation for XQuery, it has been adopted as the de-facto industry standard for querying XML data.

Due to its central nature, XPath is related to many other XML-related standards of the W3C. XQuery uses XPath for selecting nodes in tree models of XML documents. XPath expressions can be evaluated as part of XQuery queries. However, XQuery goes far beyond XPath and provides features for recombining and building new XML data. In fact, it is often compared to SQL, even though it does not yet have the same functionality as SQL for relational data. However with the recent work on the XQuery Update Facility *(XQuery, 2008)* an important step was done into this direction.

Clearly, the development of XPath and XQuery is coordinated to ensure consistency, and the editors of these standards work together closely, see *(XDM, 2007)*. This document describes almost main aspects regarding of the language and relationship with XPath 2.0. A variety of publications of the W3C provides a formal theoretical basis for XQuery and describes the relationship between XQuery, XPath and other XML-related standards, see for example *(XQuery, 2007a)*.

Since the introduction of XMLType in Oracle XMLDB, there is ongoing work in the SQL/XML committee to provide efficient XML querying capabilities in the SQL standard using XQuery. A straightforward approach referred to as the coprocessor approach, is to simply embed an XQuery process and treat the XQuery related function as a black-box – sending the queries over to the embedded processor and getting the results *(Liu, Krishnapreasad, and Arora, 2005)*.

SQL/XML defines the SQL functions XMLQuery and XMLTable as a general interface between the SQL and the XQuery language. Like other SQL/XML functions, XMLQuery and XMLTable provide developers the power and flexibility of both SQL and XML. By using these functions, developers are able to construct XML data using relational data, query relational data as if it were XML, and construct relational data from XML data. Simple examples of using the XMLQuery function (for running an XQuery query) and the XMLTable function (for running an SQL query) with XMLType data look as follows:

XMLQuery example:

```
SELECT warehouse_name,
        XMLQuery(
          'for $i in /Warehouse
           where    $i/Area > 80000
           return <Details>
                      <Docks num="{$i/Docks}"/>
                      <Rail>{if ($i/RailAccess = "Y") then "true" else "false"}
                      </Rail>
                  </Details>'
         PASSING warehouse_spec RETURNING CONTENT) big_warehouses
  FROM warehouses;
```

XMLTable example:

```
SELECT items.pos, items.itemno, items.quatity

FROM purchaseorder p,

    XMLTable(‘for $i in /PurchaseOrder//Items

                Where $i/Quantity > 200

                Return $i' passing p.pocol

                Columns pos for ordinality,

                    Itemno number path ‘ItemNo’

                    Quantity number DEFAULT 0 path ‘Quantity’

    ) items;
```

With the help of the XMLQuery and XMLTable functions, the Oracle DBMS can natively query over relational, object-relational and XMLType data. They are used for a variety of applications which incorporate lots of tasks such as product documentation and technical manuals; product specification and design documents; and web site content etc.

Common reasons for using XQuery to query XML are given in (*Oracle, 2005b)*:

- Conciseness and simplicity
- Heterogeneous queries
- Semi-structured XML data model
- XML construction

For the perspective of web application developers, the initial use of XML as a storage format is broadening into the need to query the stored XML data from XML-enabled

database. The native XQuery support within Oracle database could be another promising solution for those web application development tools.

4.5 Programming Support for XMLType Processing

As we mentioned, several industry-standard APIs that allow programmatic generate, retrieve and manipulate of XML content from SQL, Java, PL/SQL and C. Some specific implementations for XML processing for APIs implementation with Java are already proposed including SAX, DOM, JAXP, JDOM and dom4j.

4.5.1 XMLType Processing with Java

JDOM, as a typical one, is a "more intuitive, tree-based API designed just for Java. It makes no compromises to support other languages. It includes Java specific features like equals() and hashCode() methods. Where appropriate it makes free use of standard Java classes like List and String. Consequently JDOM feels a lot more intuitive to most Java programmers than DOM does" *(Harold, 2002)*. This API programming model is created by Jason Hunter who works as an independent consultant. JDOM uses JAXP and compatibly integrate with SAX and DOM APIs. It is born because of the insufficiency of DOM and SAX. It is a successful case that helps linking Java and XML in order to make them more natural fit.

Oracle XML DB already supports the Java Document Object Model (DOM) Application Program Interface (API) for XMLType. It is a generic API for both schema-based XML documents and non-schema-based XML documents so that it can handle all kinds of valid XML documents regardless of how they are stored in Oracle XML DB. This Java DOM API for XMLType can be used to construct an XMLType instance from data encoded in different character set and a set of new methods for retrieving the XML contents in the requested character set.

Oracle XML DB provides several methods for Java applications to access, manipulate and update XML data stored in a database by using JDBC. The overall Java DOM API for XMLType calling sequence description is described in Section 13 (called Java API for XMLType) of the Oracle XML DB Developer's Guide (*Oracle XML DB, 2005)*:

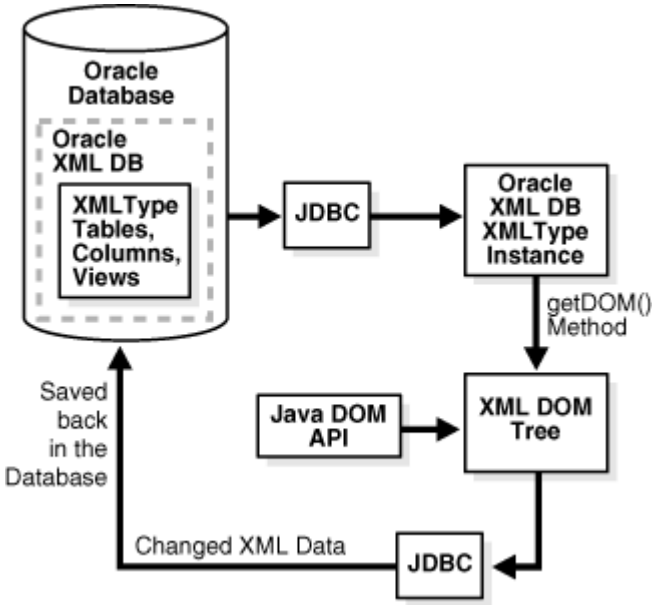


Figure 35: Java DOM API for XMLType: Calling Sequence

This methodology is basically a SQL-based approach for accessing any data in Oracle Database including XMLType data by using oracle.xdb.XMLType class and methods. It gives developers different options to achieve the goal. They can query an XMLType table to obtain a JDBC XMLType interface which supports all methods by the SQL XMLType data type; they can select XMLType data in JDBC by using certain methods in SQL including getClobVal( ), getStringVal( ) or getBlobVal(csid) and get corresponding object instance of oracle.sql.CLOB, java.lang.String or oracle.sql.BLOB in Java; or use getOPAQUE( ) call in Java prepared statement to get whole XMLType instance and construct an oracle.xdb.XMLType class out of it. A simplest example of using getObject( ) to directly get XMLType from the ResultSet in Java application shows as follow:

```
import oracle.xdb.XMLType;
...
PreparedStatement stmt =    conn.prepareStatement(
                "select e.poDoc from po_xml_tab e");
ResultSet rset = stmt.executeQuery();
while(rset.next())
{
// get the XMLType
XMLType poxml = (XMLType)rset.getObject(1);

// get the XML as a string...
String poString = poxml.getStringVal();
  }
```

4.5.2 XMLType Processing with PHP

Besides Java, PHP is another popular scripting language to develop web application. When developers work on Oracle XML DB with PHP, they can also process XML data both on the client side by using PHP's XML extensions and processing XML content inside the database. *Vasiliev (2007)* lists the following advantages of processing XML content inside a DBMS:

- Benefiting from the XML-specific memory optimizations,
- Eliminating overhead associated with parsing XML documents, and
- Reducing disk I/O operations and network traffic between the client and the database.

Vasiliev (2007) further discusses a method that developers can access relational and object-relational data from XML application by using XMLType views. The main idea is to create an XMLType view on certain table and issue queries against the XMLType view to retrieve data in XML format and manipulate XML data with internal XML tree of DomDocument instance. Furthermore, developers can convert XMLType data to String or CLOB in order to retrieve data by writing PL/SQL function into PHP. A typical example of XML processing with PHP is the following *(Vasiliev, 2007)*:

```
CREATE OR REPLACE VIEW dept_v OF XMLType WITH OBJECT ID
    (EXTRACT(OBJECT_VALUE, '/DEPARTMENT/@deptno').getNumberVal())
      AS SELECT XMLELEMENT("DEPARTMENT",
          XMLFOREST(DEPTNO, DNAME),
          (SELECT XMLELEMENT("EMPLOYEES",
                XMLAGG(
                  XMLELEMENT("EMPLOYEE",
                      XMLFOREST(EMPNO, ENAME, JOB)
                    )
                  )
                )
           FROM emp e WHERE e.deptno = d.deptno
           )
          )
        FROM dept d;


<?php
  //File: XMLFromSQL.php
  $user = 'scott';
  $pswd = 'tiger';
  $db ='(DESCRIPTION=
          (ADDRESS_LIST=
            (ADDRESS=(PROTOCOL=TCP)(HOST=localhost)(PORT=1521))
           )
```

```
            (CONNECT_DATA=(SID=orcl)(SERVER=DEDICATED))
          )';

  $sql = "SELECT value(d).GetStringVal() as RESULT FROM dept_v d
            WHERE extractValue(OBJECT_VALUE,'/DEPARTMENT/DEPTNO')=:deptno";

  $deptno=10;

  //Connect to the database and obtain info on a given department in XML format
  $conn = oci_connect($user, $pswd, $db);
  $query = oci_parse($conn, $sql);
  oci_bind_by_name($query, ":deptno", $deptno, 2);
  oci_execute($query);
  oci_fetch($query);
  $strXMLData = oci_result($query, 'RESULT');

  //Create a new DOM document and load XML into its internal XML tree
  $doc = new DOMDocument("1.0", "UTF-8");
  $doc->loadXML($strXMLData);

  //For simplicity, just print out the XML document
  print $doc->saveXML();

?>
```

4.5.3 XMLType Processing with Ruby

Similar with PHP, Ruby also provides a workaround to handle XMLType data stored in Oracle Database. Since ruby-oci8 driver does not support new XMLType data, it throws an exception about user defined types if developers try to query a table with an XMLType column. However, the Ruby-Oracle driver developer Kubo is continuously processing and temporarily recommends a working solution to deal with the problem. The workaround basically uses a normal method which returns XMLType data as CLOB which is illustrated as follow:

XMLTYPE can be retrieved as CLOB as following:

```
----------------------------------------------------------------
require 'oci8'

def select_as_csv(conn, sqltext)
  conn.exec(sqltext) do |row|
    row.collect! do |col|
      case col
      when OCI8::LOB
        col.read()
```

```
        else
          col
        end
      end
      puts(row.join(','))
    end
end

conn = OCI8.new('ruby', 'oci8')

#conn.exec("CREATE TABLE po_tab OF XMLTYPE")
#conn.exec("CREATE TABLE po_xml_tab(poid NUMBER(10), poDoc XMLTYPE)")

xml1 = '<root><key>John</key><val>Smith</val></root>'
xml2 = '<root><key>bar</key><val></val></root>'

conn.exec('insert into po_tab values(:1)', xml1)
conn.exec('insert into po_tab values(:1)', xml2)
select_as_csv(conn, 'select XMLTYPE.getClobVal(value(x)) from po_tab x')

conn.exec('insert into po_xml_tab values(:1, :2)', 1, xml1);
conn.exec('insert into po_xml_tab values(:1, :2)', 2, xml2);
select_as_csv(conn, 'select poid, XMLTYPE.getClobVal(podoc) from
po_xml_tab')

conn.rollback
---------------------------------------------------------------

To insert a XML value, bind as a String.
To select a XML value, use XMLType.getClobVal(CLOBColumn).
```

All those programming languages mentioned above could be used for developing web-based applications as scripting languages for either services/operations or database communication. They could provide web application developers more possible alternatives other than native PL/SQL or XQuery to handle XMLType data stored within Oracle Database. Although they use some workarounds and temporary solutions rather than XML-friendly and native solutions, they may give developers more flexibility and simplicity.

5 Implementing Workarounds

5.1 Oracle Application Express with PL/SQL

As discussed above, Oracle APEX does not natively support XMLType data. The CREATE TABLE wizard does not allow application developers to choose an XMLType column at the first position. And also, the wizard generated forms that use automated DML are not XMLType aware as well. However, developers can use a workaround to incorporate uses a data source with XML data and processes XML data through custom SQL query or PL/SQL processes within web application generated.

The key point of this workaround to handle XMLType column is using custom region sources or page processes to realize data fetching and manipulating. In order to show specific data from stored XMLType data, developers can create a custom SQL report region to retrieve values from XML document and displayed as report columns rather than only “datatype” as we see in problem statement section above. The proposed SQL query used in the prototype looks as follows:

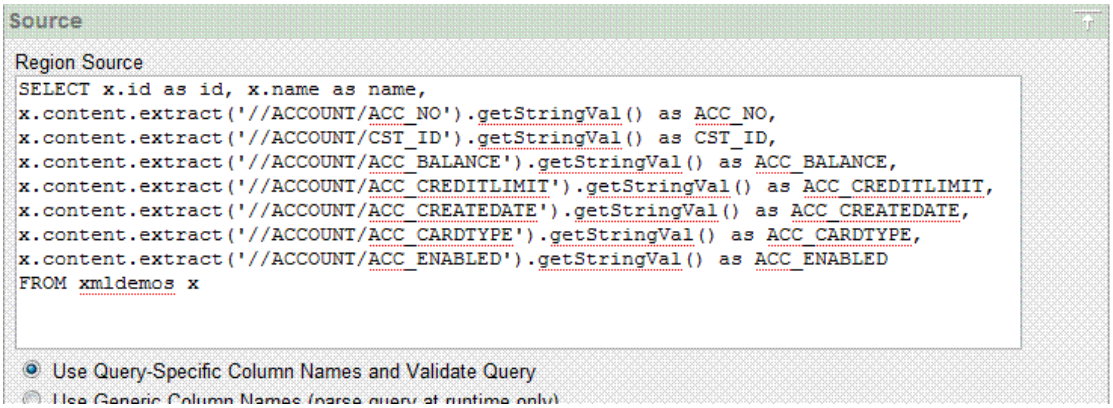


Figure 36: Region source editor for report in Oracle APEX

This is only one display structure of SQL report on this table. The key is using XMLType API functions (such as getStringVal(), getClobVal() or getNumVal) to extract XML element(s) and get the value within them to display on the report. The SQL region source gives us the following report:

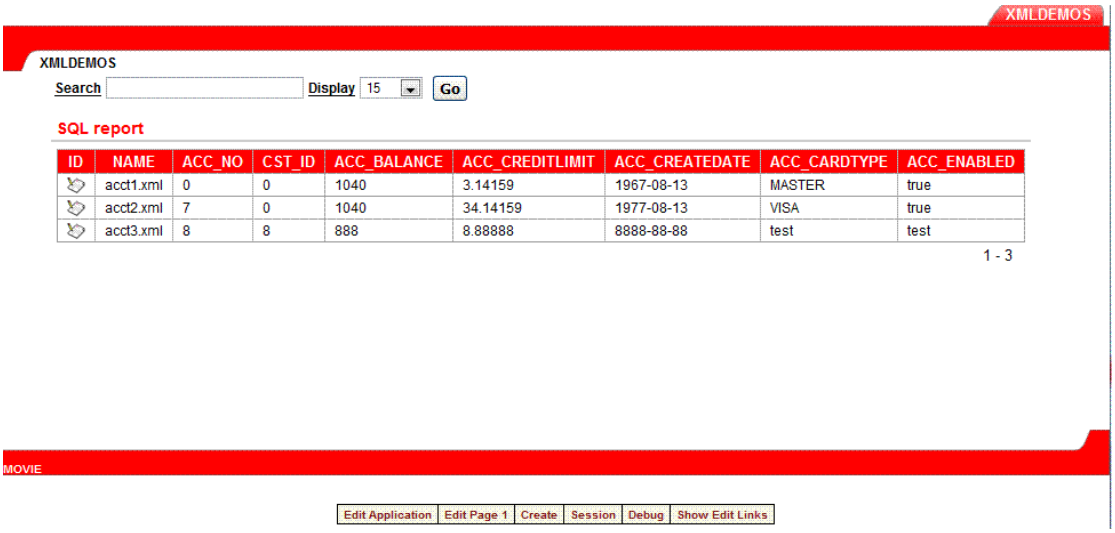


Figure 37: XML demo table page in Oracle APEX (modified)

Within the record form page, the Automatic Row (DML) Processing Process ("Fetch Row from XMLDEMOS") of page rendering section is generated automatically through application wizard. That pre-defined page process cannot handle XMLType data. What developers need to do is to create custom PL/SQL processes for row fetching and manipulating to substitute for the default processes. The proposed PL/SQL fetch process is made as follows:

Source

* Process [Download Source]

```
declare
xmldoc varchar2(4000) := null;
begin
for a1 in ( select * from xmldemos where id = :P2_ID)
loop
xmldoc:= xmltype.getClobVal(a1.content);
:P2_NAME := a1.name;
:P2_CONTENT := xmldoc;
:P2_ACC_NO := a1.content.extract('//ACCOUNT/ACC_NO/text()').getStringVal();
:P2_CST_ID := a1.content.extract('//ACCOUNT/CST_ID/text()').getStringVal();
:P2_ACC_BALANCE := a1.content.extract('//ACCOUNT/ACC_BALANCE/text()').getStringVal();
:P2_ACC_CREDITLIMIT:= a1.content.extract('//ACCOUNT/ACC_CREDITLIMIT/text()').getStringVal();
:P2_ACC_CREATEDATE:= a1.content.extract('//ACCOUNT/ACC_CREATEDATE/text()').getStringVal();
:P2_ACC_CARDTYPE:= a1.content.extract('//ACCOUNT/ACC_CARDTYPE/text()').getStringVal();
:P2_ACC_ENABLED:= a1.content.extract('//ACCOUNT/ACC_ENABLED/text()').getStringVal();
end loop;
end;
```

Figure 38: Source code for retrieval of XML demo data in Oracle APEX

This chunk of PL/SQL procedure is mainly used to retrieve all data from xmldemotable table which includes XMLType column as we created above. The XMLType data is retrieved by using getClobVal pre-defined function of xmltype and stored into a temporary variable xmldoc. It holds the whole XML document data which is finally assigned to Oracle APEX item variable :P2_CONTENT. We also retrieve all values within document elements and separately listed in the form. Like auto-generated "Fetch Row from XMLDEMOS" process, this new process is set to be executed after header and before region rendered. Besides, the source of page item P2_CONTENT must be set to item variable :P2_CONTENT rather than database

column. The result form with XMLType data loaded shows as follows in Oracle APEX:

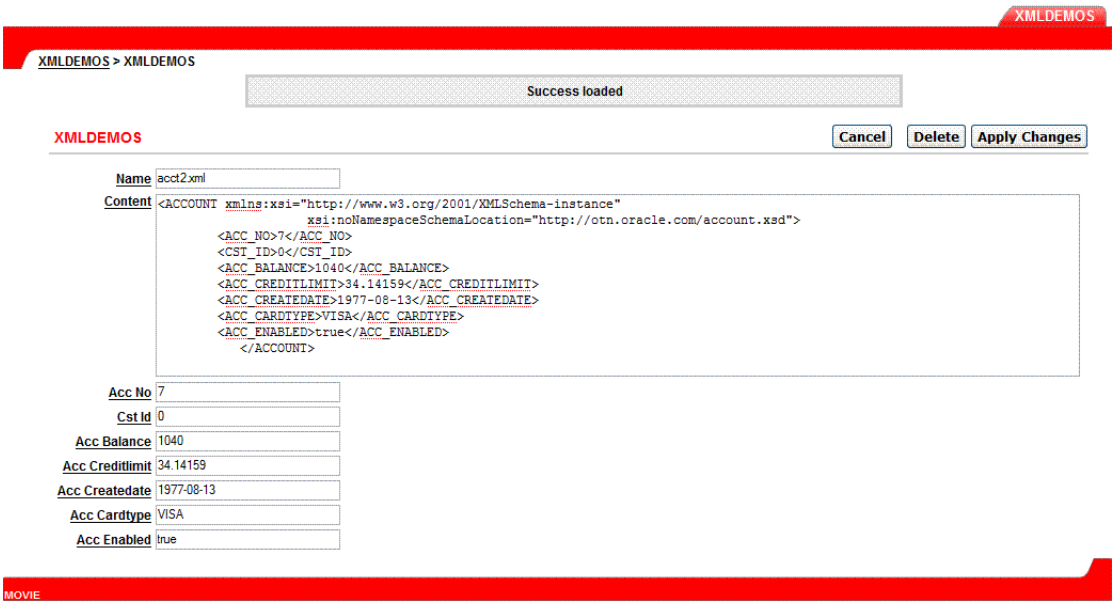
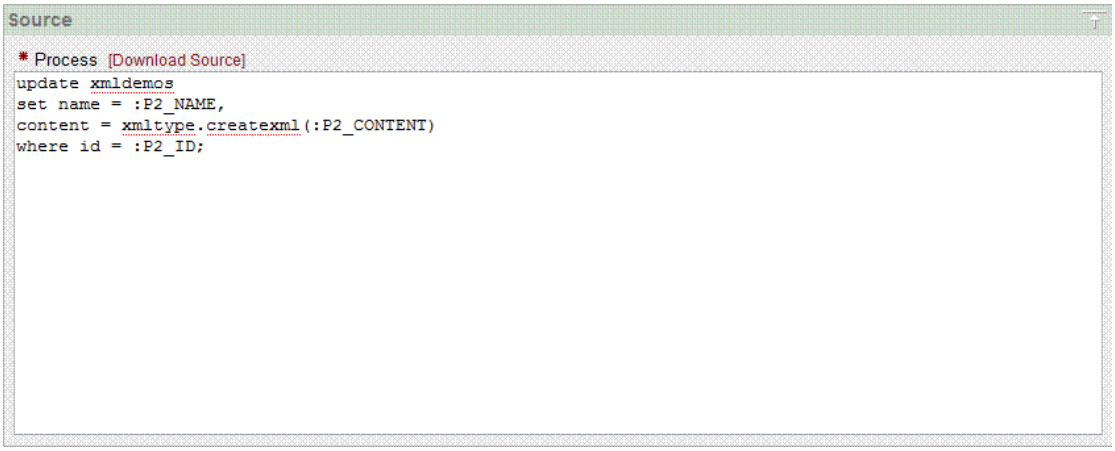


Figure 39: Form for XML demo data in Oracle APEX (modified)

The update functionality within the application also needs to be modified in order to successfully update XMLType column data within the table. As we did for XML data retrieval, new possible custom processes under page processing section need to be generated as follows:



Source

* Process [Download Source]

```
update xmldemos
set name = :P2_NAME,
content = UPDATEXML(UPDATEXML(UPDATEXML(UPDATEXML(UPDATEXML(UPDATEXML(content,
'//ACCOUNT/ACC_NO/text()', :P2_ACC_NO), '//ACCOUNT/CST_ID/text()', :P2_CST_ID),
'//ACCOUNT/ACC_BALANCE/text()', :P2_ACC_BALANCE), '//ACCOUNT/ACC_CREDITLIMIT/text()',
:P2_ACC_CREDITLIMIT), '//ACCOUNT/ACC_CREATEDATE/text()', :P2_ACC_CREATEDATE),
'//ACCOUNT/ACC_CARDTYPE/text()', :P2_ACC_CARDTYPE), '//ACCOUNT/ACC_ENABLED/text()', :P2_ACC_ENABLED)
where id = :P2_ID;
```

Figure 40: Source code for updating the XML demo data in Oracle APEX

The main idea of updating XMLType column data is using SQL/XML functions such as CREATEXML() or UPDATEXML(). By using createxml() function, the whole xml document stored in XMLType column is updated with the page item in the form (e.g. :P2_CONTENT) while updatexml() function provides mechanism that allows developers to update specific element value within XML document and return an updated xmltype object. All these methods provide web application developers abilities to either retrieving or manipulating XMLType data from databases.

5.2 WaveMaker with JAVA

Due to the unsupported datatype exception error, WaveMaker cannot handle XMLType data from XML-enabled databases initially either as shown below:

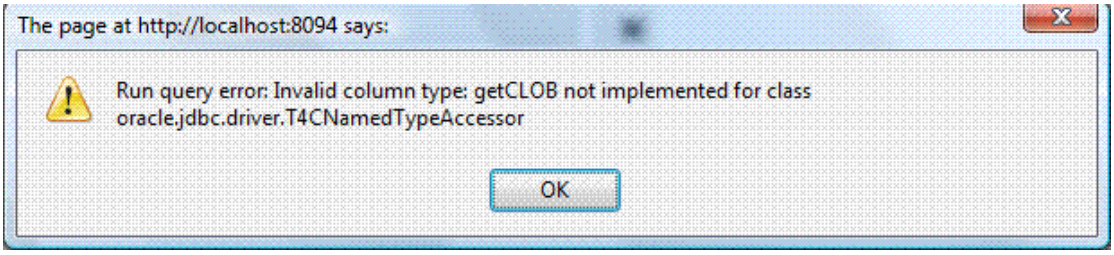


Figure 41: Exception error when retrieving XML demo data in WaveMaker

Since WaveMaker reduced the number of supported programming languages for custom services from three (Python, Java, PHP) of ActiveGrid to two (Java and JavaScript) of WaveMaker. It doesn't give developers many choices to choose from in order to find out a workaround by using programming languages.

The key point for WaveMaker to handle XMLType data is to use Java custom service as an extra data service for the special column within Oracle database. A data service interface with all necessary methods for retrieving or manipulating and Java Class for specific object (e.g. xmldemos) is automatically generated after developer imports database as a new data model. Some typical "CRUD" data related methods are insertXmldemos, getXmldemosById, updateXmldemos, deleteXmldemos,

getXmldemosList, getXmldemosCount. However, CLOB object is not currently supported in the current JDBC driver. By investigating into the source code of data service interface generated, we found the input object instance or returned object instance within these methods in Xmldemos interface class are defined as Xmldemos type whose class is stored in data folder. Each imported database table has three types of files to define properties and functions. They are hibernate-mapping xml file, java class file and database query xml definition. It is impossible for developers to modify the data service manager class of WaveMaker to change the data model mechanisms but after we modify all CLOB type for content XMLType column data to String type in project service class and hibernate xml file, we finally get the record list on the page as follows:

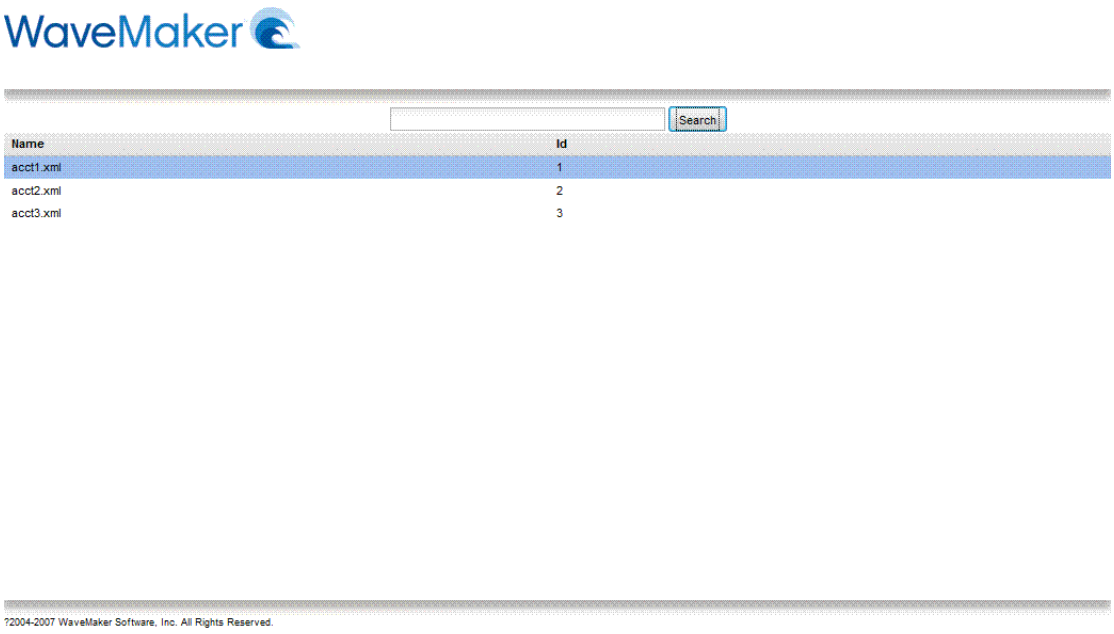


Figure 42: XML demo table page in WaveMaker (modified 1)

owever, String type cannot hold XMLType data retrieved from database so that the XMLType column “content” data still cannot be correctly retrieved and shown in the list page. Furthermore, developers are not capable of processing or transforming XMLType column data but only retrieving the whole instance of Xmldemos class because the current version WaveMaker only allows developers to modify condition part of the database query. That restricts the possibility for developers to use “query-level” functions to deal with XMLType data as we did in Oracle APEX. The workaround developers could use to achieve the goal is using a custom service which is separate from Xmldemos class auto-generated by WaveMaker. It looks as follows:

```
import com.activegrid.runtime.data.DataServiceManager;
import com.activegrid.runtime.data.DataServiceManagerAccess;
import com.activegrid.runtime.data.QueryOptions;
import com.activegrid.runtime.data.TaskManager;
import com.activegrid.runtime.service.RuntimeServiceAccess;
```

```
import com.activegrid.runtime.AGRuntime;
import com.wavemaker.xmldemostable.Xmldemostable;
import com.wavemaker.xmldemostable.data.Xmldemos;
import org.hibernate.Session;
import org.hibernate.Query;
import java.util.*;
import java.math.BigDecimal;

public class MyRetrieveXmldemosClass {
    public List<XmlWrapper> getXmlTypeValue() {
        Xmldemostable xmldemostable = (Xmldemostable)
        AGRuntime.getInstance().getService("xmldemostable");
        xmldemostable.begin();
        try {
            Session session = xmldemostable.getDataServiceManager().getSession();
            Query q = session.createSQLQuery("select x.id, x.name, x.content.getStringVal() from
            xmldemos x");
            List<XmlWrapper> rtn = new ArrayList<XmlWrapper>();
            List<Object[]> l = q.list();
            for (Object[] row : l) {
                rtn.add(new XmlWrapper((BigDecimal)row[0], (String)row[1], (String)row[2]));
            }
            return rtn;
        } finally {
            xmldemostable.rollback();
        }
    }
    public static class XmlWrapper {
        private BigDecimal id;
        private String name;
        private String content;
        public XmlWrapper(BigDecimal id, String name, String content) {
            this.id = id;
            this.name = name;
            this.content = content;
        }
        public BigDecimal getId() {
            return id;
        }
        public String getName() {
            return name;
        }
        public String getContent() {
            return content;
```

```
        }
    }
}
```

This custom Java service class demonstrates how to obtain the XMLType value by reusing the DataServiceManager Session class within WaveMaker and retrieve the String format of XMLTyep data by passing custom SQL query to Oracle database. First of all, we need a custom XmlWrapper to hold the returned database result object because the auto-generated Xmldemos object regards XMLType as CLOB variable which causes the problem for retrieving XMLType data. The only thing different in the inner class is the data type for content which is String. The main method getXmlTypeValue in this class mainly reuses DataServiceManager Session class and use createSQLQuery to pass our custom SQL query. The returned list object is finally stored into XmlWrapper type list and returned as the method result. Finally we use "DataGrid" widget item to hold different columns and bind corresponding fields from the returned dataset of our custom Java service. The final result is show as follows:

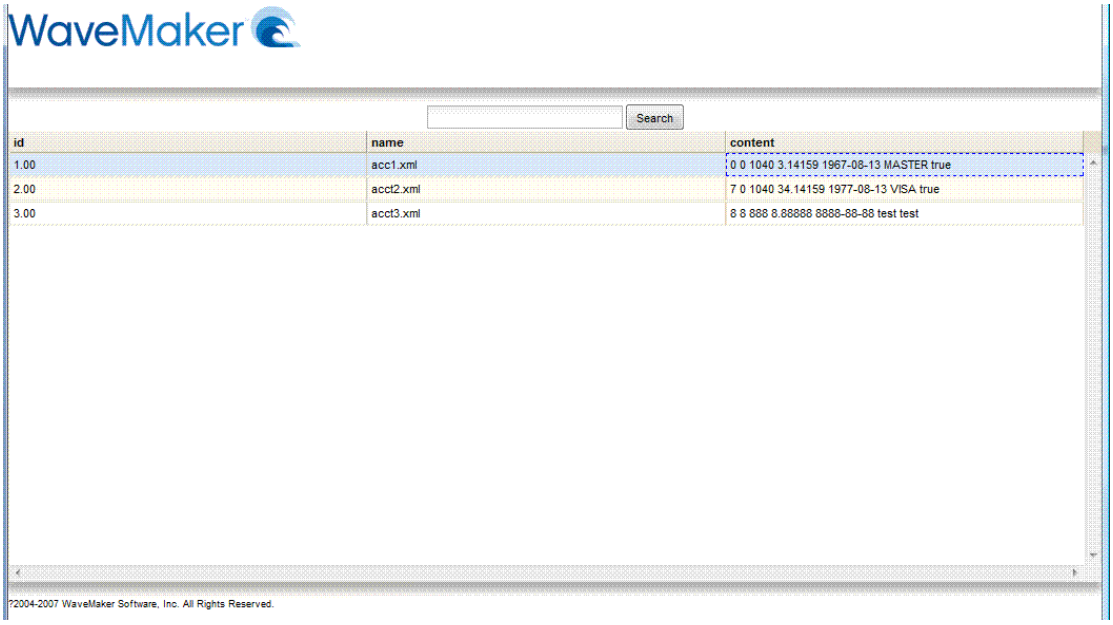


Figure 43: XML demo table page in WaveMaker (modified 2)

The returned object from method getXmlTypeValue() of MyRetrieveXmldemosClass class includes all database columns value collection. We can also modify the SQL query within createSQLQuery method to retrieve certain XML element/element values as we did in Oracle APEX. The possible result is:

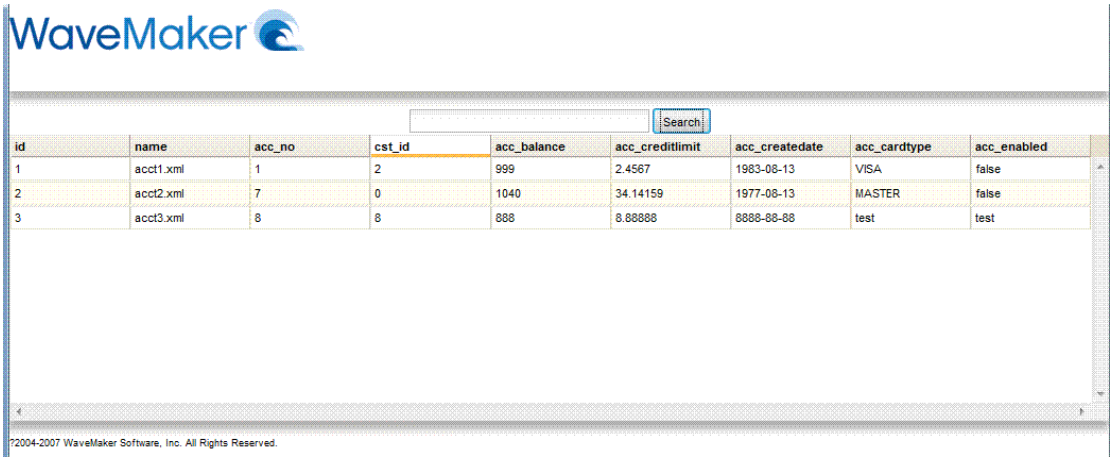


Figure 44: XML demo table page in WaveMaker (modified 2)

In order to manipulate on the content of the XML document, we follows the same routine processes to generate a form which holds the specific record and allows end-users to update the chosen record. The difference from normal form is that we manually created an text area form element and binds the corresponding result data value from getXmlTypeValue() function. And also, we can also get individual element value from the String value of XML document. The final edit form looks as follows:

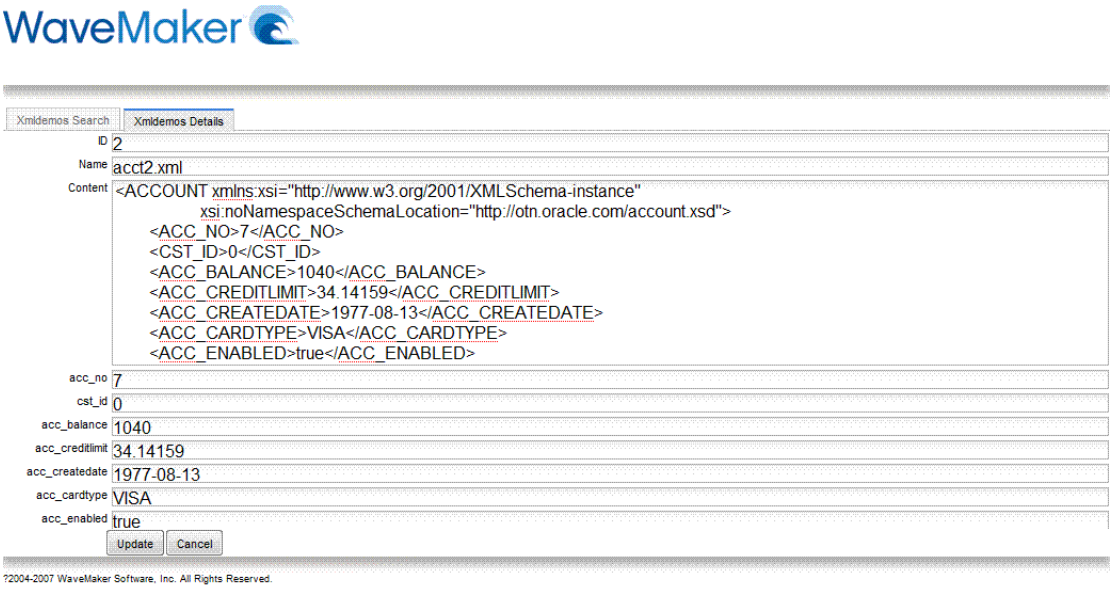


Figure 45: Form for the XML demo data in WaveMaker

The last task we have to accomplish is to update XMLType column data through the edit form. As we did for XMLType data retrieval, developers can create another update method updateXmlTypeValue(Xmldemos xmldemos) within RetrieveXmldemos class for updating XMLType data. The method is declared as follows:

```
import com.wavemaker.xmldemostable.data.Xmldemos;
public void updateXmlTypeValue(Xmldemos xmldemos) {
```

```
    Xmldemostable xmldemostable = (Xmldemostable)
    AGRuntime.getInstance().getService("xmldemostable");
    xmldemostable.begin();
     try {
          Session session = xmldemostable.getDataServiceManager().getSession();
         Query q = session.createSQLQuery("update xmldemos set name = " + xmldemos.getName +
         " content = " + xmldemos.getContent);
     } finally {
          xmldemostable.rollback();
     }
}
```

This new method mainly uses the same workaround as getXmlTypeValue method which make use of createSQLQuery() method belonging to WaveMaker dataservicemanager session object. Besides, we can also use purely Java implementation regardless of pre-defined WaveMaker runtime classes which is only suitable for advanced programmers as follows:

```
import com.wavemaker.xmldemosnew2.data.Xmldemonew;
  public static void updateXmlcontent (Xmldemos xmldemos) throws Exception    {


    try {
       OraclePreparedStatement pstmt = (OraclePreparedStatement) conn.prepareStatement("update
xmldemos set content = XMLType(?)");
       String xmlString = xmldemos.getContent();
XMLType xmldat = XMLType.createXML(conn, xmlString);
pstmt.setObject(1, xmldat);
pstmt.execute();
    } catch (SQLException SQLe) {
         if (pstmt != null) {
           pstmt.close();
           throw SQLe;
         }
    }
}
```

In this new method "updateXmlcontent", an Xmldemos instance is passed as a parameter. The "xmlString" variable holds the data to be entered into the database and is retrieved through invoking getContent method belonging to Xmldemos instance. The rest is to use XMLtype function createXML to create XMLType data from String data and finally store the generated object into database. Besides, we also need to setup all bindings within the service call including all variables of Xmldemos object and assign this new function to "update" button in the end. All processes above are main ideas developers may use to deal with XMLType column data in WaveMaker.

5.3 Ruby on Rails with ActiveRecord

The Ruby/Oracle connection driver ruby-oci8 is a base level Oracle driver for Ruby developed by Kubo Takehiro, see *(Kubo, 2008)*. It is already installed after Ruby on Rails installation. If one wants DBI style access, then this is all one needs. However, I have found out that this connection driver cannot handle XMLType data stored in Oracle databases. ActiveRecord-oracle-adapter (ActiveRecord-oci-adapter is the old version name) turns to be another option. ActiveRecord-oracle-adapter is a Rails gem, which depends upon ruby-oci8, and provides Rails/ActiveRecord support especially for Oracle. The gem installation of ActiveRecord-oracle-adapter is as follows:

gem install ActiveRecord-oracle-adapter –source http://gems.rubyonrails.org

Since we attempt to use the new Oracle connection driver, we also have to change the config/database.yml database settings from oci to oracle. After all those pre-steps for web application development, we finally get the basic web application on Xmldemos table with XMLType column.

Figure 46: XML demo table page in Ruby on Rails (modified 1)

Although the useful scripts (script/generate scaffold) automatically generate basic model (CRUD), controllers, and view files according to the MVC model for the part of the application that use the corresponding table specified in ruby command. Developers may also want to have further processing on XMLType data such as retrieve element value of sub-element rather than only the whole document content. As we introduced above, Ruby on Rails is Model-View-Controller (MVC) web-application development framework. Rails uses the ActiveRecord database abstraction classes to work with database tables. This is simply defined within project model file:

```
class Xmldemo < ActiveRecord::Base
end
```

Active Record class ActiveRecord::Base has method called connection() which returns valid connection to connected database using config/database.yml database settings. Return value of connection() method is an object of database connection adapter class. Some useful methods of its object are execute, insert, update, delete, begin_db_transaction, commit_db_transaction and rollback_db_transaction. We can use these methods to manually execute our SQL query as we did in Oracle APEX or WaveMaker in order to obtain the returned resultset variable because sometimes it is

more convenient to work with plain SQL queries (without any model classes) for specific database operations. We use “index” controller function as the example to show possible solutions for XMLType.

Old “index” controller status

```
  # GET /xmldemos
  # GET /xmldemos.xml
  def index
    @xmldemos = Xmldemo.find(:all)

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @xmldemos }
    end
  end
```

New “index” controller status

```
  # GET /xmldemos
  # GET /xmldemos.xml
  def index
    xmldemoconn = Xmldemo.connection()
    xmldemoconn.execute “SET autocommit=0”
    xmldemoconn.begin_db_transaction
    @xmldemos = Xmldemo.execute(“select x.id, x.name,
x.content.getStringVal() from xmldemos x”)
    Xmldemoconn.commit_db_transaction

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @xmldemos }
    end
  end
```

Alternatively, developers can also reuse methods pre-defined in ActiveRecord::Base class such as find, find_by_sql, create, update, delete etc. In the default project controller file, we have already seen some of them and we just need to modify those existing functions to achieve the goal. An example to show how to use find_by_sql looks like as follow:

```
  # GET /xmldemos
  # GET /xmldemos.xml
  def index
    @xmldemos = Xmldemo.find_by_sql("select x.id, x.name,
```

```
x.content.extract('//ACCOUNT/ACC_NO/text()').getStringVal() as acc_no
from xmldemos x")

    respond_to do |format|
      format.html # index.html.erb
      format.xml  { render :xml => @xmldemos }
    end
  end
```

Both of these two approaches still take advantage of pre-defined functions in Rails and direct SQL query usage when we attempt to meet the requirement of retrieving XMLType column data from databases. One can use a programming language level solution or workaround to achieve the goal, such as the proposal to use the Oracle connection adapter ruby-oci8 *(Kubo, 2008)* that was mentioned above. However, the native query-level solution should provide a more efficient way to solve this problem. The rest of the listing page development are just put returned variable into page HTML code. The final listing page looks as follows:

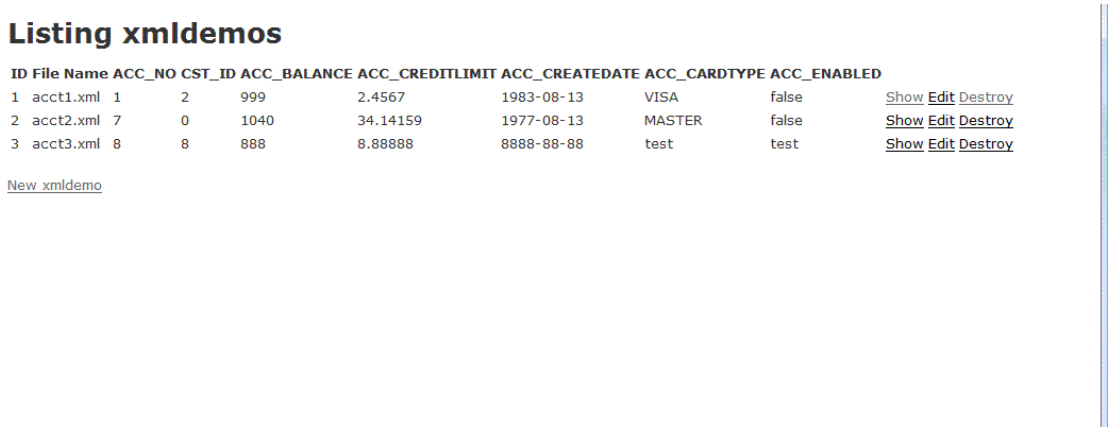


Figure 47: XML demo table page in Ruby on Rails (modified 2)

The edit form is created using the similar method as index listing page. The only difference is that the SQL query needs to use params[:id] to get specific record and show attribute values in the form elements. Code looks like:

```
  # GET /xmldemos/1/edit
  def edit
    @xmldemo = Xmldemo.find_by_sql("select id, name,
content.getStringVal() as content from xmldemos where id = ?",
params[:id])
  end
```

The edit form looks as follows:

Figure 48: Form for the XML demo data in Ruby on Rails

The final XMLType data manipulation is also easy to achieve. Since ActiveRecord-oracle-adapter has already provided developers a working interface for updating all columns within database tables includes XMLType data. We can directly reuse the update function to fulfil the manipulation task. If developers want to store those XML document element values into table, they only need to declare essential variables and reuse update_attributes function to achieve it without any other modifications.

6 Comparison and Empirical Study

6.1 Native Query Languages vs. Programming Languages

The obvious advantage of ease-of-use, existing programming languages usually offer expressive power, allowing complex ideas to be expressed in a few lines of code. They also leverage your existing knowledge, so you can be productive right away". As Micheal states, programming languages such as Java, Ruby, Python or PHP provide developers more functional power when meeting with variety of developing requirements. Java and JavaScript gives WaveMaker developers more opportunities to realize certain sophisticated functionalities; Ruby, as the programming language in Ruby on Rails, make full use of Rails development framework. Even when developers attempts to deal with XMLType data within database, they also plays an essential role for either creating custom data service or modifying existing pre-defined. Developers who have enough knowledge on these programming languages can have solutions or workarounds to handle XMLType data.

In contrast, domain-specific languages let developers work with domain concepts (like XMLType data) directly. For example, most general-purpose programming languages treat XML as any other API, instead of as a first-class part of the language. Instead of providing operators for constructing and navigating XML directly, you have to access it through an API layer. Just as text manipulation is easier in Perl than in, say, Fortran, So a single line of an XML query language like PL/SQL, XSLT or XQuery can accomplish the equivalent of hundreds of lines of Java, Ruby, Python, or some other general-purpose programming languages. In Oracle APEX, the whole development process does not require any programming languages or APIs supports. Developers only need to use native PL/SQL or XQuery in order for creating custom data processing or XML document/elements/element values manipulation.

As for as performance goes, there are three reasons that domain languages usually outperform general-purpose languages. One is that domain languages are usually optimized for tasks common to that domain. General-purpose programming languages have to perform well on a wide range of tasks, while XML query languages only have to perform well on a narrow set of common XML tasks. This focus may limit their applicability, but often yields superior performance. Since certain native query languages such as PL/SQL or XQuery are especially designed for querying data from database. Compared with transforming XMLType data and data mapping by using programming languages within WaveMaker or Rails framework, Oracle APEX purely using query language PL/SQL or XQuery takes less time and less code to retrieve, insert, update or delete XMLType data stored in Oracle database.

Another is that the general-purpose language will use an API of some sort for working with that domain. The abstraction layer provided by that API almost always hides the internal data structures from software using the API. Information hiding is a great

benefit to program design, but can introduce overhead and by definition prevents you from manipulating the underlying data directly. In contrast, a query language is the abstraction. When the query is executed, it has full access to all aspects of its own internal data structures.

However, the main reason XML query languages can outperform general-purpose programming languages is that they are less constrained. A programming language usually has to do exactly what you tell it to do, in the order you specified. Without extensive analysis, temporary intermediate results must be computed exactly as described. For instance, to successfully create a new custom data service using programming languages in WaveMaker or Ruby on Rails framework, developers must be clear about the whole processing sequence and essential classes import. On the other hand, in Oracle APEX developers only need to consider query-level solution when they meet the XMLType column data requirements. That relatively saves a large amount of overall development time.

The point is clear: Query languages enjoy certain advantages over traditional, general-purpose programming languages. These advantages are illustrated in the success of SQL for accessing relational data, and the successes of many other domain-specific “little languages,” from regular expressions to shell scripts to graphics libraries. In this case, Oracle APEX that is mainly take advantage of PL/SQL and XQuery stands on a fairly higher position than WaveMaker and Ruby on Rails framework.

6.2 XMLType Processing Workarounds

With the help of integrated native query languages PL/SQL or XQuery, developers working on Oracle APEX to develop web applications do not need to have any professional software development knowledge or general-purpose programming language skills. The Oracle APEX web-based development environment provides both entry-level application developers and professional developers all essential control flow and functionalities in the way developers can easily modify. It separates three groups of page section and all types of components within each section. The workarounds for processing XMLType data mainly focuses on “Page Rendering” and “Page Processing” sections. The wizard-based application development style eliminates unnecessary fatal errors when developers generate any item, page region or process. As we showed above, native PL/SQL Query for XMLType data retrieving or manipulation can the only tool when we create either page region or page process. All XMLType processing happens within the query only so that developers can ignore any other unrelated issues such as page layout definitions or basic functionalities. Even for items used in application to hold retrieved XML document element value, developers can also imply create them and modify attributes belonging to it. Generally speaking, the whole development process takes less time than other development tools because of its integrated features in Oracle Database and is suitable for all levels of developers. However, the obvious disadvantage is database dependence. The efficient native

development can only be implemented on Oracle database which makes it somehow only suitable for Oracle application development.

WaveMaker, as an inheritor of ActiveGrid, uses cut-edge technologies such as AJAX, Hibernate mapping library, Spring and Dojo Toolkit and wide range of database support such as Oracle, DB2 and MySQL. It provides developers a real enjoyable developing environment which allows developers to develop a web-based application by using Drag-and-Drop and innovated features. All basic developing functionalities reside in the “blackbox” of WaveMaker, which allows developers to build application with minimal code. When XMLType column data introduced, WaveMaker pre-defined dataservice manager and task manager cannot initially handle it.

Since WaveMaker is still under development, XMLType column data is not yet fully supported as a common data type in database tables. Developers have to create custom vender-specific data services to handle it. WaveMaker provides developers two optional scripting languages for these custom services which are Java and JavaScript. That initially makes it hard for low-level web application developers who are not familiar with these two programming languages to quickly and correctly implement their own services. The workaround implementation involves pre-defined data service manager and runtime classes in WaveMaker and reuses declared methods in order to retrieve and manipulate XMLType data from database. As a solution of using programming language to handle XMLType data, the efficiency of it strongly depends on the developers’ familiarity on the specific language. Although the AJAX developing environment gives developers an enjoyable development, the “black box” backend functionalities enclosure results in inconvenience and inefficiency when developers attempt to make customized service or modifications. That would be the limitation compared with other two type of development tools.

Compared with the two development tools above, Ruby on Rails as a web application development framework gives developers a pure-Ruby development environment. To go live, all one needs to add is a database and a web server. Rails works with a variety of web servers and database systems. For DBMS one can use MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2, and a number of further DBMS. Ruby on Rails does not provide developers an “easy-to-use” web-based user interface like Oracle APEX or WaveMaker. In particular for less experiences developers this can be a challenge when combining application components.

The Model-View-Control pattern separates different layers of a web application. The data processing definitions resides in “Control” part. The default Ruby-oci8 Oracle database connection adapter has no support for Oracle XMLType columns. The adapter developer gave out a workaround to return string format of XMLType data. Furthermore, the ActiveRecord module has new connection adapter (ActiveRecord-oracle-adapter) recently updated has already supported XMLType data. Since ActiveRecord as Rails model communicating with database provides powerful a class (ActiveRecord::base) with all essential methods to cope with XMLType data.

Developers have several options for processing XMLType by using specific pre-defined methods (e.g. find_by_sql/connection().execute) of the ActiveRecord::base class. Compared with the other two web application development tools, Ruby on Rails is somehow more pure and flexible even though the installation is not so convenient for entry-level developers. The good news is that there is an installation package with all components integrated which also saves a lot of time when setting up the tool.

7 Conclusions and Recommendations

With emergence of the Web 2.0 idea more and more web-based application development tools are introduced and used for efficient application development. Web 2.0 applications require highly interactive user interfaces and means for application integration and adaptation. Application developers need to be supported by powerful, yet easy-to-use development tools to meet these requirements.

Developers now have many choices of web-based application development tools such as model-driven development tool (e.g. Oracle APEX), integrated development environment packages (e.g. ActiveGrid/WaveMaker) and development framework on pure programming language (e.g. Ruby on Rails). They all have their own features, advantages and disadvantages. Some of them provides entry-level developers easy-to-use web-based application development interface for communicating with integrated database such as Oracle APEX; some of them such as WaveMaker uses industry-standard cut-edge technologies to provide both lower level developers and higher level developers efficient and enjoyable development environment for developing web application on different types of databases; and some of them use combination of pure programming language and developing framework such as Ruby on Rails to enhance flexibility for either scripting or database connection in application development. Different levels of users may have different choices on these tools depending on their developing skills and application requirements.

After the new XMLType data type introduced in most popular databases as XML-enabled databases and widely used XML documents within web-based applications. It requires web application development tools to work on XML data stored in back-end XML-enabled database table XMLType column. However, these web application development tools do not originally have supports on XMLType data. Most of them cannot correctly show XMLType column data in application and there are no such functionalities to retrieve certain XML document sub-element values for manipulation; some of them even cannot generate application if XMLType data occurs in the data model for application. Developers can use of respective workarounds or proposed solutions in order to successfully access, retrieve and manipulate XMLType data stored in underlying databases in web-based applications created with the development tools under discussion.

Within all workarounds to deal with XMLType, native Query languages let developers work with domain concepts directly. Other than those general-purpose programming languages used in some web-based application development tools, developers are able to use simple query languages like PL/SQL or XQuery to provide operators for constructing and navigating XML data directly without need to access it through any API layer. This kind of solution used in model-driven development tool Oracle APEX is suitable for entry-level developers such as end-users or staff members. It saves these developers unnecessary time on advanced programming but still realize the

XMLType data processing through native query.

With the help of programming language workaround as scripting languages for custom service definition, both lower-level and professional developers can find a solution to handle XMLType data. In integrated development environment package WaveMaker, developers can also generate the application with functionalities to handle XMLType data by using its easy-to-use AJAX panel with drag-and-drop feature. The only necessary tweak regarding XMLType is to create custom service as a data source extension to process XMLType data stored in XML-enabled database. That remains the basic application developing process but still has advanced data service especially programmed for XMLType data processing. And also, it supports most popular XML-enabled databases used as data model for application development. That gives web application development on XMLType more flexibility and compatibility on WaveMaker.

As the last type of development tool introduced, Ruby on Rails developing framework provides developers high flexibility in many aspects. Since newer Rails framework provides powerful data module (ActiveRecord) to handle data model, developers can reuse certain existing methods within new Rails to process XMLType data without creating any new data services. Furthermore, different kinds of alternative database connection adapters can be implemented in order for different purposes and Ruby on Rails supports almost all popular XML-enabled databases. Developers only need to find out suitable database connection adapter used for Ruby on Rails for XMLType data processing which gives developers easiest way to solve the unsupported XMLType datatype problem.

At last, whatever development tool developers choose for XMLType data processing ability and whatever workarounds and solutions available for XMLType data processing, it must be suitable based on the level of developers and development requirements. These three types of development tools have their own solutions to handle XMLType data. In order to achieve more efficient web application development on XML-enabled databases, developers have to decide all necessary criteria before they choose the most suitable web application development tools.

8 References

Abiteboul, S., Buneman, P., and Suciu, D. (1999): *Data on the Web: From relations to semistructured data and XML*, Morgan Kaufmann.

ActiveGrid (2006): *Introducing Enterprise Web 2.0*, ActiveGrid White Paper.

Baker, D., Romano, A., and Winters, T. (2008): *Oracle Database 2 Day + Application Express Developer's Guide*, Release 3.1.2, Oracle.

Bansal, V. and Alam, A. (2001): Study and Comparison of Techniques to Efficiently Store and Retrieve XML Data, Project report, Duke University.

Baxley, B. (2003): What is a Web Application, *Boxes and Arrows*, Available online at http://www.baxleydesign.com/publications/articles/baa_web_app_defined.html

Beyer, K. and Cochrane, R. (2006): DB2 goes hybrid - Integrating native XML and XQuery with relational data and SQL, *IBM Systems Journal*, Vol. 45, No. 2.

Bhattacharya, S. (2006): Provide key features in a Web-based application, *IBM developerWorks*, Available online at http://www.ibm.com/developerworks/library/wa-abilities.html

Booch, G. (2001): The architecture of Web applications, *IBM developerWorks*, Available online at http://www.ibm.com/developerworks/ibm/library/it-booch_web/

Bourret, R. (2005): *Going native: Use cases for native XML databases*, Available online at http://www.rpbourret.com/xml/UseCases.htm

Bourret, R. (2005a): *XML Database Products*, Available online at http://www.rpbourret.com/xml/XMLDatabaseProds.htm

Chamberlin, D. (2002): XQuery - An XML query language, *IBM Systems Journal*, Vol. 41, No. 4.

Chen, H. and Tompa, F.W. (2003): Set-at-a-time access to XML through DOM, *ACM Symposium on Document Engineering*, pp. 225-233

Cohen, G. (2008): Where did the auto-generated Web pages go? *WaveMaker Studio Community*. Available online at http://dev.wavemaker.com/forums/?q=node/1296

Constantin, L. (2007): WaveMaker Offers WYSIWYG Ajax Development, *Softpedia*, Available online at http://news.softpedia.com/news/WaveMaker-Offers-WYSIWYG-Ajax-Development-92088.shtml

Davis, I. (2005): Talis, Web 2.0 and All That, *Internet Alchemy* blog. Available online

at http://internetalchemy.org/2005/07/talis-web-20-and-all-that/

Doyle, B. and Lopes, C.V. (2008): *Survey of Technologies for Web Application Development*, 43 pp., Available online at http://arxiv.org/abs/0801.2618v1

Feuerstein, S. (2005): *Oracle PL/SQL Programming*, 4th edition. O'Reilly.

Fraternali, P. (1999): Tools and approaches for developing data-intensive Web applications: A survey, *ACM Computing Surveys*, Vol. 31, No. 3, pp. 227-263.

Garrett, J.J. (2005): *AJAX: A New Approach to Web Applications*, Available online at http://www.adaptivepath.com/ideas/essays/archives/000385.php

Handy, A. (2007): Former ActiveGrid Makes Waves With App Studio: Framework update launched with new name and new Web development suite, *SD Times* blog. Available online at http://www.sdtimes.com/article/story-20071215-03.html

Harold, E.R. (2002): *Processing XML with Java*, Addison-Wesley.

Herrán, L.B. (2006): *Evaluation of the Ruby on Rails Framwork*, Master thesis, University of Gent, 94pp.

Hogan, B.P. (2006): Serving Multiple Rails Applications on Windows with Apache and Mongrel, Revision 1.0, Available online at http://www.napcsweb.com/howto/rails/deployment/RailsWithApacheAndMongrel.pdf

Klaene, M. (2004): Building Web Applications with Oracle HTML DB, Part 1, *Developer.com*, Available online at http://www.developer.com/db/article.php/3384201

Klaene, M. (2004a): Building Web Applications with Oracle HTML DB, Part 2, *Developer.com*, Available online at http://www.developer.com/db/article.php/3399331

Ku, K.I., Kang, S.J., Chung, M., Kim, W.Y., and Choi, W. (2008): Evaluation and Optimization of ActiveRecord to implement the Personalized Software Service Platform, *IEEE International Conference on Advanced Communication Technology*, pp. 1763-1766.

Kubo, T. (2008): ruby-oci8 1.0.1, *Softpedia*, Available online at http://linux.softpedia.com/get/Programming/Libraries/ruby-oci8-39127.shtml/

Laddad, R. (2000): XML APIs for Databases, *Java World*. Available online at http://www.javaworld.com/javaworld/jw-01-2000/jw-01-dbxml_p.html

Liotta, M., and Preimesberger, C. (2003): Native XML databases resolve XML document retrieval issues, *Builder.com*, Available online at http://builder.com.com/5100-6388_14-1051795.html

Liu, Z.H., Krishnaprasad, M., and Arora, V. (2005): Native XQuery Proocessing in Oracle XMLDB, *ACM SIGMoD Conference*, pp 828-833.

Møller, A. and Schwartzbach, M. (2006): *XML and Web Technologies*, Addison Wesley.

Murugesan, S., Deshpande, Y., Hansen, S., and Ginige, A. (2001): Web Engineering: A New Discipline for Development of Web-based Systems, Web Engineering, *Lecture Notes in Computer Science*, vol. 2016, pp. 3-13.

Nicola, M., and Van der Linden, B. (2005): Native XML Support in DB2 Universal Database, *VLDB Conference*, pp. 1164-1174.

Obasanjo, D. (2001): *An exploration of XML in database management systems*, Available online at http://www.25hoursaday.com/StoringAndQueryingXML.html

Oracle (2005): Integrating Oracle Application Express with Oracle Application Server 10g, Oracle White Paper.

Oracle (2005a): Oracle Database 10g Release 2 - XML DB Technical Overview, Oracle White Paper.

Oracle (2005b): XML Query (XQuery) Support in Oracle Database 10g Release 2, Oracle White Paper.

Oracle APEX (2008): Oracle Application Express, Release 3.1.2, Release Notes, Oracle.

Oracle XML DB (2003): *Oracle XML DB Utilities Package*, Oracle Technology Network.

Oracle XML DB (2005): *Oracle XML DB Developer's Guide*, Oracle Technology Network.

O'Reilly, T. (2005): Web 2.0 – Compact Definition, *O'Reilly Radar* blog. Available online at http://radar.oreilly.com/archives/2005/10/web_20_compact_definition.html

O'Reilly, T. (2005a): *What is Web 2.0? - Design Patterns and Business Models for the Next Generation of Software*, Available online at http://www.oreilly.com/go/web2/

Pettit, S. (2001): *Anatomy of a Web Application: Security Consideration*, Sanctum.

Sklar, L., and Rosen, R. (2003): *Web Application Architecture*, Wiley.

Smith, K. (2006): Simplifying AJAX-Style Web Development, *IEEE Computer*, Vol. 39, No. 5, pp. 98-101.

Staken, K. (2001): Introduction to Native XML Databases, *XML.com*, Available

online at http://www.xml.com/pub/a/2001/10/31/nativexmldb.html

Tatarinov, I., Ives, Z.G., Halevy, A.Y., and Weld, D.S. (2001): Updating XML, *ACM SIGMoD Conference*, pp. 413-424.

Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J., and Zhang, C. (2002): Storing and querying ordered XML using a relational database system, *ACM SIGMoD Conference*, pp. 204-215.

Thuraisingham, B. (2002): *XML Databases and the Semantic Web*, CRC Press.

Vasiliev, Y. (2007): Building Database-Driven PHP Applications on Oracle XML DB, *Oracle Technology Network*, Available online at http://www.oracle.com/technology/pub/articles/vasiliev_xmldb_php.html

Vosloo, I. and Kourie, D.G. (2008): Server-centric Web frameworks: An overview, *ACM Computing Surveys*, Vol. 40, No. 2, pp. 1-33.

WaveMaker (2007): *WaveMaker Studio User Guide*, Version 4.0, WaveMaker.

Williams, J. (2007). *Rails Solutions: Ruby on Rails Made Easy*, Apress/friends of ED, distributed by Springer.

XDM (2007): XQuery 1.0 and XPath 2.0 Data Model (XDM), W3C Recommendation, Available online at http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/

XML (2006): Extensible Markup Language (XML) 1.0, Fourth Edition, W3C Recommendation, Available online at http://www.w3.org/TR/2006/REC-xml-20060816/

XML (2006a): XML 1.1, Second Edition, W3C Recommendation, Available online at http://www.w3.org/TR/2006/REC-xml11-20060816/

XML Schema (2004): XML Schema Part 0, Primer, Second Edition, W3C Recommendation, Available online at http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/

XPath (2007): XML Path Language (XPath) 2.0, W3C Recommendation, Available online at http://www.w3.org/TR/2007/REC-xpath20-20070123/

XQuery (2007): XQuery 1.0 - An XML Query Language, W3C Recommendation, Available online at http://www.w3.org/TR/2007/REC-xquery-20070123/

XQuery (2007a): XQuery 1.0 and XPath 2.0 Formal Semantics, W3C Recommendation, Available online at http://www.w3.org/TR/2007/REC-xquery-semantics-20070123/

XQuery (2008): XQuery Update Facility 1.0, W3C Candidate Recommendation,

Available online at http://www.w3.org/TR/2008/CR-xquery-update-10-20080801/

XSLT (2007): XSL Transformations (XSLT) 2.0, W3C Recommendation, Available online at http://www.w3.org/TR/2007/REC-xslt20-20070123/