Connecting XForms to Databases

An Extension to the XForms Markup Language

Mikko Honkala
Nokia Research Center
mikko.honkala@nokia.com

Oskari Koskimies
Nokia Research Center
oskari.koskimies@nokia.com

Markku Laine
Helsinki University of Technology
mplaine@cc.hut.fi

1. INTRODUCTION

Authoring of web applications is typically complex because the author has to master many programming languages and paradigms. The user interface of a modern web application is authored in HTML, JavaScript and CSS. The server side, however, is typically authored in a language like PHP, Java or Ruby, and database access uses either direct SQL commands or a database-to-objects mapping. The concepts and programming paradigms are vastly different. This client-server divide makes authoring of a new web application a challenging task which is best left to experts. Even when experts are used, there are still problems: business logic is divided between client and server, even limited offline support is difficult to accomplish, and multi-user support such as synchronization and transactions need to be considered separately for each application.

One way of bridging the client-server gap is to unify the client and server programming under a single model. This single model can be based on server-side concepts, such as automatically generating web forms from a database schema description, or it can be based on client-side concepts, such as extending JavaScript with database access functions. In general it is not possible to delegate the authoring of a user interface to a machine, so using client-side concepts as a basis is preferable. However, just gluing together client and server side concepts (such as the JavaScript-SQL integration just mentioned) will not provide us with a single, easy-to-use authoring paradigm. What we need is way to seamlessly integrate the database, the application logic and the user interface under a single concept.

The most difficult thing is database access. For the integration to be seamless, database access must be integrated to the authoring language itself, instead of being just an add-on library. One way of doing this would be to export database contents as programming language objects, similar to Ruby on Rails [11] or Hibernate [12]. However, although programming languages are flexible, they are inherently complex and thus relying on a programming language limits the audience of the solution. We believe that a declarative authoring language, where the author states his intent rather than the implementation details, would be usable by a wider audience.

The XForms form description language is such a declarative language. Furthermore, the XForms language is designed for authoring forms that edit an XML document. When the database is modeled as an XML document, XForms natively supports accessing and modifying the database. XForms is thus uniquely amenable to database integration.

We have designed extensions to the XForms language that enable a natural integration with databases. The extensions rely on a generic server-side component for implementing the database integration and can be compiled back to XForms 1.1 compliant markup.

2. RELATED WORK

Our proposal is a so-called end-to-end solution for authoring Web applications. There are other similar approaches. Google Web Toolkit (GWT) [9] is a recent approach that tries to unify Web development by allowing both the client and the server-side to be programmed using Java programming language. In contrast to GWT, which is purely procedural, our approach is completely declarative, allowing for non-programmers and interoperable authoring tools.

There are also other proposals for integrating XForms and XQuery, such as [4]. While this is in line with XFormsDB, it is lacking one important aspect: easy-to-use updates of the data edited by the form. In our proposal it is solved with synchronization of XML fragments.

3. PROBLEM STATEMENT

While the basic problem has already been explained, it is worth explaining some of our assumptions and defining the exact problem we aim to solve.

We assume that declarative languages are easier for non-programmers than imperative languages, and we have chosen the XForms language as the starting point since it is the forms part of XHTML 2.0. The other logical alternative would have been HTML, but HTML relies on JavaScript for client-side logic which makes it unsuitable for declarative authoring. XForms is also interesting because it is well suited for implementation on mobile devices (e.g., [5], [1]).

XForms was designed to replace complex and hard-to-maintain JavaScript code with declarative markup. Its purpose is to replace HTML and JavaScript with something that provides the same functionality in a better way. However, XForms does not aspire to provide functionality beyond that what is offered by HTML and JavaScript. The XForms programming model assumes a separate server-side component, and XForms is not concerned with how the server-side component is authored. This is where our extension comes in. Our basic research question is the following:

How can we extend XForms so that it is possible even for non-programmers to author useful web applications entirely in XForms?

It is worth noting that we do not aim to support authoring of every possible type of web application. Instead the purpose is to allow people to easily author the most typical web applications. The server-side components in typical web applications all have one thing in common: they provide the client-side component with access to a database. In fact, almost all of the functionality the server-side component typically provides is connected to the database access in one way or another: access control and user authentication define what parts of the database the user is allowed to access, session management provides the context and authorization for database queries, and multi-user support is implemented by database transactions. We therefore can reformulate our research question as follows:

How can we add database connectivity to XForms in such a way that access control, authentication, session management and multi-user support is preserved, while keeping the language simple enough for non-programmers to use?

Our proposed solution is described in the following sections.

4. SOLUTION DESIGN

We first evaluated the feasibility of allowing the XForms form access the contents of a remote XML database in the same way as it access the local XML data, i.e. via XPath. However, usage scenario analysis quickly showed that XPath was less than ideal for general database searches, because it is very hard to make queries with join operations in XPath. For that, XQuery is needed. Unfortunately XQuery is prohibitively heavy to implement, especially in the mobile environment. We therefore do not consider it feasible to require an XQuery implementation on the client side. Instead we have implemented a generic server-side component which provides XQuery access to the database.

In our solution, the XForms client can access database data in two ways. Firstly, it can ask the server to evaluate an XQuery and return the resulting document. Because the result can contain elements not present in the original data, changes to the resulting document cannot be reliably mapped back to the original data. Therefore, the client can access the result data but cannot submit changes back to the server.

Secondly, the client can request a document fragment from the server. The fragment is identified by an XPath expression pointing to the root element of the fragment. Once the fragment has been returned, the client can modify it and submit back changes. The server uses an XML synchronization engine [2] to merge the changes with any concurrent changes from other clients, and in the absence of conflicts, commits the merged changes to the database. If there is a conflict, the submission of changes fails.

For security reasons, the queries are never seen by the client in their original form. Instead, the server replaces queries with opaque reference IDs which the client uses to execute a query. This prevents a malicious client from rewriting the queries. The client can, however, parameterize the queries if allowed by the author.

It is worth noting that while XQuery can be complicated, it is a superset of XPath so the user can rely on XPath alone as long as he does not need the extra features XQuery provides. Furthermore, when the database contents are small enough to fit in client memory (e.g. certain collaborative applications), the form can access and modify database contents just like instance data.

From database operation point of view, the requesting of a document fragment corresponds to beginning a transaction and the submission of changes corresponds to committing a transaction. If a transaction spans several fragments, the fragments need to be grouped together so that no changes are committed to the database if even one of the fragment submissions fails.

We considered implementing native support for our language extensions in an open source XForms browser such as [5], but this would have limited the practical usefulness of our system. Instead we chose to implement our language extensions as a transformation from XFormsDB to standard XForms. Thus, any standard-compliant XForms client[1] will be compatible with our system. This includes server-side solutions such as [3], [7].

5. THE XFORMSDB LANGUAGE

The XFormsDB language is a pure superset of XForms 1.1 [10], which is still a work in progress at W3C. It contains database-related additions, but all user interaction is authored in pure XForms. While the main goal has been to allow hand-authoring of multi-user web applications, another design goal has been the possibility to implement the language as a transformation, whose result can be fed into a XForms 1.1 – capable client. Compared to XForms 1.0, XForms 1.1 adds features for XML fragment handling and repetitive actions, which are essential in building database applications.

5.1 Element xformsdb:instance

XFormsDB defines a new element xformsdb:instance, which contains a database query.

[1] Currently we require that the XForms 1.1 (in Last Call at the time of writing) extensions to the <insert> action are supported in the client.

This query, which can be written in XQuery or XPath, is to be executed at the server when a corresponding submission is dispatched. The actual query can be initiated multiple times and at any point in the lifetime of the form and is triggered by, similarily to plain XForms, the xforms:submission element.

5.2 Element xformsdb:query

This element contains the query to be executed in the server. The query is either inline in this element or referenced with an URL. It contains the following child elements and attributes:

The attribute *datasrc* is an optional attribute, which points to an id in the application-level configuration file. By default, the first data source is used.

The attribute *doc* is an optional attribute, which can be used to change the context node of the query into a specific document in the collection of the data source. By default, the whole collection is used as the context node.

The element *expression* contains the server-side XQuery (select) or XPath (select and update). The query can be either a reference in the @src attribute (a relative URL resolved in the context of the form in the server), or inline in this element.

The element *var* allows of the change the external variables for the query. The attribute name has to mach a variable name, while the value is read from the text content of this element.

5.3 Extensions to element xforms:submission

XFormsDB extends the element xforms:submission with the following attributes:

The attribute expressiontype determines the type of the query to be run. It can have two values: “select” or “update”. By default, “select” is used.

The attribute queryinstance is an id-reference to a xformsdb:instance element, which holds the query to be executed.

The processing of xforms:submission is similar to pure XForms 1.1 submission processing. The XFormsDB specific processing is triggered by the existence of the queryinstance attribute.

5.4 Errors: Event xformsdb-*query*-error

This event is dispatched in response to an xml database error, for instance loss of database connectivity, or problem in synchronization. It can be catched by normal XForms event handlers using XML Events. The target is the submission element and the event bubbles and is cancelable. In addition to the event, the following structure is appended into the root node of the xformsdb:instance.

```
<error>
  <code>4093</code>
  <message>
   Failed to establish a connection to the
   data source.
  </message>
</error>
```

Listing 2: An error message from a database operation

5.5 Example: Server-side queries defined in XQuery

Server-side queries are expressed in XQuery, which allows range of different kinds of queries from simple node selections to complex generated queries using database joins. For instance, the example in Listing 3 defines a simple XQuery, which contains no parameters, and whose result is stored into the instance with the id “instance”.

```
<xformsdb:instance id="update-queryinstance">
  <query xmlns="" datasrc="exist" doc="helloworld.xml">
    <expression>
      /root/helloworld/message
    </expression>
  </query>
</xformsdb:instance>
<xformsdb:submission id="select4update-querysubmission" replace="instance"
  instance="instance" queryinstance="update-queryinstance"
  expressiontype="select">
</xformsdb:submission>
<xformsdb:submission id="update-querysubmission" replace="instance"
  instance="instance" queryinstance="update-queryinstance" expressiontype="update">
</xformsdb:submission>
```

Listing 1: Definition of an updatable XML fragment

```
<xformsdb:instance id="select-
queryinstance">
  <query xmlns="" datasrc="exist"
    doc="helloworld.xml">
    <expression>
     for $helloworld in
      /root/helloworld/message
     return $helloworld
    </expression>
  </query>
</xformsdb:instance>
<xforms:instance id="instance"/>
<xformsdb:submission id="select-
querysubmission" replace="instance"
instance="instance" queryinstance="select-
queryinstance" expressiontype="select" />
```

Listing 3: Definition of a server-side query

5.6 Example: synchronized and updatable XML fragments in XPath

In order to change parts of the database, the form can retrieve an XML fragment, edit it, and submit changes, which will get merged in to the database. From the language point of view, this is done similarly to pure XQuery queries, except that two submission elements are used: one for selecting the fragment, and the second for submitting the changes to the synchronization process. For example, the code example in Listing 1 will define an updatable XML fragment.

6. PROTOTYPE IMPLEMENTATION

As a proof of concept, the XFormsDB framework was implemented. The prototype implementation has been built using J2EE technologies, consisting of several open-source libraries. The development environment included X-Smiles [5], eXist [6], and Tomcat running on Windows XP.

6.1 Architecture

The architecture of the XFormsDB framework (cf. Figure 1) can be divided into four logical tiers: presentation tier, application server tier, integration service tier, and data tier.

The *Presentation Tier* is responsible for transforming XFormsDB files to XHTML + XForms 1.1 compliant markup meaning that web browsers using the XFormsDB framework are required to support XForms 1.1. However, in the future this restriction can be removed by adding a server-side XForms 1.1 processor, e.g. Orbeon Forms [7], which translates XForms to HTML, CSS, and JavaScript that web browsers understand.

The *Application Server Tier* is in charge of managing queries and XML query results, as well as merging XML query results when an update expression is performed.

The *Integration Service Tier* provides an interface for the application server tier to query different types of data sources using the XQuery or XPath language. The data source is queried via virtual XML view which is constructed with an appropriate mapping strategy. Currently, the XFormsDB framework supports only XML-based data sources, but by using a middleware, e.g. DataDirect XQuery [8], support for

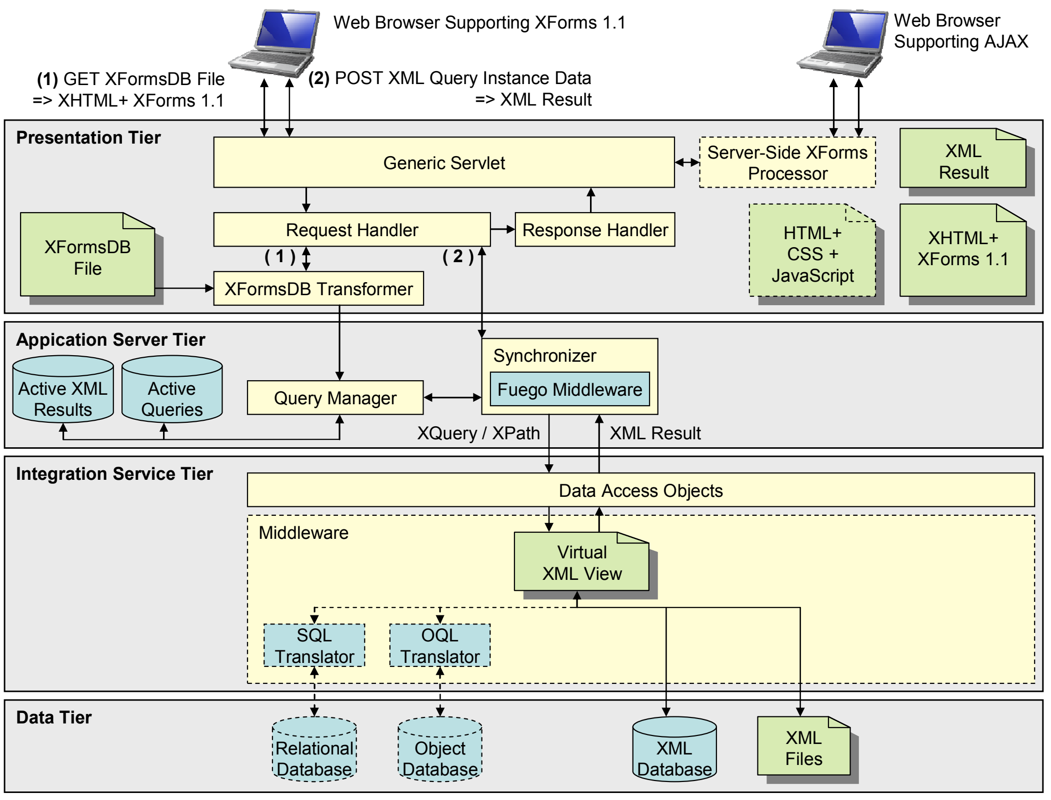

Figure 1: Architecture of the XFormsDB framework (future work marked with dash)

other types of data sources could be added as well.

The *Data Tier* stores application data in a persistent store, such as an XML database or a relational database.

6.2 Component Descriptions

The main components of the XFormsDB framework in Figure 1 are described below.

- **Generic Servlet:** Receives all incoming requests; acts as a centralized gateway to the server.
- **Request Handler:** Handles the request according to the HTTP method (GET or POST), file extension, and URL information.
- **XFormsDB Transformer:** Parses through a requested file authored in the XFormsDB language and collects all database-related additions. After the parsing has been done, the collected query instances are stored into the session for future reference. Finally, the transformer produces an XHTML + XForms 1.1 compliant document where the collected query instances are replaced by reference IDs.
- **Query Manager:** Manages query instances and XML query results stored into the session.
- **Synchronizer:** Parses through a posted query instance and retrieves the authored query instance from the session by means of the supplied reference ID. In case of a select expression, the query is executed and XML result is returned to the client. Again, if the same query instance is used for both selecting (for fetching data to be updated later) and updating, then in addition to the process described above, the XML result is also stored into the session. This procedure makes it possible to execute update expressions as described in detail in section 4.
- **Data Access Objects:** Defines a common interface for the application server tier to query different types of data sources using the XQuery or XPath language.
- **Response Handler:** Writes response content according to the given input parameters.

7. CONCLUSION AND FUTURE WORK

We have implemented an initial prototype of a system that enables entire web applications to be authored easily in a single document, using the XFormsDB markup language which is based on XForms and provides convenient database access in addition to standard XForms functionality. Our extensions are implemented by transforming the XFormsDB document to a standard-compliant XForms document, which works together with a generic server-side database access component.

Possible future work includes:

- Integrating our prototype with a server-side XForms processor
- Developing a web-based authoring environment
- Further simplifying the XFormsDB syntax
- Integration to web services in a manner consistent with the database integration
- Improving transaction support
- Extending XForms with more convenient notation for common use cases

8. REFERENCES

[1] PicoForms Mobile. Software. Available at: http://www.picoforms.com/

[2] Tancred Lindholm: XML Three-way Merge as a Reconciliation Engine for Mobile Data. In *Third ACM International Workshop on Data Engineering for Wireless and Mobile Access*, September 2003.

[3] Chiba. Software. Available at: http://chiba.sourceforge.net/

[4] Mark Birbeck, Application of XML Access: XForms and XQuery, via REST, Talk. Available at: http://www.w3c.rl.ac.uk/pastevents/XML_Access_Languages/Mark/xforms-xquery.html

[5] P. Vuorimaa, T. Ropponen, N. von Knorring, and M. Honkala. A Java based XML browser for consumer devices. In *The 17th ACM Symposium on Applied Computing*, Madrid, Spain, March 2002.

[6] eXist (online). eXist – Open Source Native XML Database: Overview. Available at: http://exist.sourceforge.net

[7] Orbeon Forms. Software. Available at: http://www.orbeon.com/

[8] DataDirect Technologies (online). DataDirect XQuery: Overview. Available at: http://www.datadirect.com/products/xquery/index.ssp

[9] Google Web Toolkit (GWT). Software. Available at: http://code.google.com/webtoolkit/

[10] J. Boyer (ed.), XForms 1.1, W3C Working Draft 22 February 2007. Available at: http://www.w3.org/TR/xforms11/

[11] Ruby on Rails. Software. Available at: http://www.rubyonrails.org/

[12] Hibernate. Software. Available at: http://www.hibernate.org/

(All web references cited April 20, 2007)