XFormsDB: A Declarative Web Application Framework

Markku Laine, Denis Shestakov, and Petri Vuorimaa

Department of Media Technology, Aalto University
P.O. Box 15500, FI-00076 Aalto, Finland
{markku.laine,denis.shestakov,petri.vuorimaa}@aalto.fi

Abstract. Most Web applications utilize a three-tier architecture, in which the presentation, application logic, and data management are implemented as separate tiers. The disadvantage of this popular approach is that it usually requires expertise in multiple programming languages and paradigms as well as data models used in each tier. A single expert rarely masters all the technologies involved. In this demonstration, we give an overview of the XFormsDB framework that allows developers to implement entire Web applications using only markup languages. The framework is based on the XForms markup language and our server-side extensions. We demonstrate the functionality of the framework using a simple blog application as an example.

Keywords: Web Framework, Web Application, Web Development, Declarative Language, XForms, XFormsDB.

1 Introduction

Highly interactive data-driven Web applications enrich the Web user experience. However, their development is complex because developers need not only to know multiple systems, frameworks, best practices, and languages, but also to deal with their conceptual dissimilarities [1]. Indeed, both imperative (e.g., Java and JavaScript) and declarative (e.g., CSS, HTML, and SQL) languages are often used together when implementing Web applications.

To provide support for complete Web application development (i.e., client-side and server-side application logic, client-server communication, and interaction), the client-side (presentation tier) and server-side (logic and data tiers) programming can be done under a single model. This architectural approach simplifies the development process, and particularly reduces the skill set required from a developer. For instance, Google Web Toolkit (GWT)[1] realizes a server-side approach, in which a general-purpose programming language (i.e., object-oriented imperative Java) is used to author not only the server-side application logic but also a Web application user interface. Similarly, a client-side programming language (e.g., XForms [2]) can be extended with server-side functionalities to cover all three tiers of a Web application. An extensive comparison of frameworks realizing different *tier-expanding* architectural approaches can be found in [3].

[1] Google Web Toolkit (GWT), http://developers.google.com/web-toolkit/

M. Brambilla, T. Tokuda, and R. Tolksdorf (Eds.): ICWE 2012, LNCS 7387, pp. 477–480, 2012.

In this demonstration, we overview XFormsDB [4], a tier-expanding Web application framework based on the XForms markup language and our server-side extensions. We describe and demonstrate its functionality by showcasing a full-fledged Web application utilizing the framework.

The advantages of the XFormsDB framework are as follows. First, as the XML data model is used on all three tiers, developers can avoid complex mappings between different data models. Second, at a minimum, developers only need to learn one markup language (i.e., XHTML 2.0) and few XForms server-side extensions. Third, all the core languages are declarative, and thus are more preferable to people with limited programming skills (particularly, to Web user interface developers, who are already familiar with declarative XHTML and CSS) [5]. Fourth, the framework is extensible: on the presentation tier, JavaScript can add animations and interactivity; on the logic tier, XQuery extension functions can extend server-side application logic; and on the data tier, XQuery can complement the more limited XPath language.

2 XFormsDB: Language and Framework Implementation

The standard XForms [2] offers only client-side functionality. Our XFormsDB markup language extends XForms with common server-side and database-related functionalities, allowing developers to implement all three tiers (i.e., presentation, logic, and data) of a Web application using only markup languages. The design of XFormsDB addresses the following general requirements of a Web application: *persistent storage*; *error handling*; *session management and security*; *modularity*; *state maintenance*; *authentication, authorization, and access control*; *uniform syntax and processing model*; and *extensible architecture* [4].

Listing 1 illustrates an example of the XFormsDB extension syntax[2]. This code snippet can be included into the `head` part of an XHTML document to provide an inclusion and authorization functionality.

```
<xformsdb:include resource="../xinc/meta.xinc"/>
<xformsdb:secview>
  <xforms:model>
    <xforms:load resource="../login.xformsdb"
      ev:event="xforms-ready"/>
  </xforms:model>
</xformsdb:secview>
<xformsdb:secview roles="admin">...
</xformsdb:secview>
```

Listing 1. XFormsDB code example

[2] For space and readability reasons, the source code (available at `http://tinyurl.com/xformsdb-blog-sc`) may differ from the code snippets shown in Listings 1-4.

As a Proof-of-Concept, we implemented the XFormsDB framework, a generic platform for developing and hosting Web applications based on the XForms markup language and our server-side extensions. The architecture of the framework is given in [4] and the framework itself is available at `http://code.google.com/p/xformsdb/`.

3 XFormsDB: Demo Description

Next, we showcase XFormsDB with a simple blog application. The application and its source code are available at `http://testbed.tml.hut.fi/blog` and `http://tinyurl.com/xformsdb-blog-sc`, respectively. The blog application has two main user interfaces, one for end users and the other for administrators. The user interfaces look and feel like any other modern Web application, i.e., they give a fast response to user inputs and remain responsive while submitted requests are being processed on the server side.

The internal architecture of the application separates the code on each Web page into the three logical tiers: data, logic, and presentation. We walk through, step by step, how a query is defined and submitted to a server, and then executed against the data stored in a database. The example continues by showing how the query result is sent back to the client, and finally ends up being displayed in the user interface.

Data Tier. The database stores published blog posts and comments in a single XML document (`blog.xml`). The hierarchical document structure has three levels: the `root` element followed by a series of blog `post` child elements, each including `comment` elements. All `post` and `comment` elements have unique identifiers. Listing 2 shows an XPath expression, which selects one blog post—defined by an external variable *$id*—and extracts its comments from the database.

Logic Tier. We divide the code responsible for the application logic between the client and the server on each Web page. We define the client-side application logic using standard XForms, whereas our XForms extensions are responsible for the application logic on the server side. Listing 3 shows a *query* command, which uses the XPath expression of Listing 2. The code snippet in Listing 4 submits the *query* command to the server, where it is securely executed against the `blog.xml` document stored in the database. For triggering the submission, the standard XForms `send` action is used. After a successful submission, the *query* result extracted from the database is stored in an XForms `instance` element, whose original content is replaced with the extracted data.

Presentation Tier. The blog user interface controls (e.g., input and output) are bound to the data in XForms `instance` elements. The user interfaces are updated every time the data in XForms `instance` elements changes, such as in the case when *query* results are received from the server.

```
/root/blog/posts/post[ @id = $id ]/comments
```

Listing 2. XPath expression to extract the comments of a selected blog post

```
<xformsdb:instance id="select-and-update-comments">
  <xformsdb:query datasrc="exist-db" doc="blog.xml">
    <xformsdb:expression resource=
      "xpath/select_and_update_comments.xpath" />
    <xformsdb:xmlns prefix="xformsdb"
      uri="http://www.tml.tkk.fi/2007/xformsdb" />
    <xformsdb:var name="id" />
  </xformsdb:query>
</xformsdb:instance>
```

Listing 3. Definition of a *query* command

```
<xformsdb:submission id="sub-select-comments"
  replace="instance" instance="comments"
  requestinstance="select-and-update-comments"
  expressiontype="select">
  <xforms:action ev:event="xforms-submit-done">...
  </xforms:action>
  <xforms:action ev:event="xformsdb-request-error">...
  </xforms:action>
</xformsdb:submission>
```

Listing 4. Definition of a *query* command submission

4 Conclusions

In this demonstration, we overviewed XFormsDB, a comprehensive Web application framework based on the XForms markup language and our server-side extensions. We demonstrated the functionality of the framework using a simple blog application as an example. The application looks and feels like any modern Web application. In addition, it runs on all modern Web browsers.

References

1. Mikkonen, T., Taivalsaari, A.: Web Applications — Spaghetti Code for the 21st Century. Technical Report SMLI TR-2007-166, Sun Microsystems (2007)
2. Boyer, J.: XForms 1.1. W3C Recommendation, `http://www.w3.org/TR/xforms/`
3. Laine, M., Shestakov, D., Litvinova, E., Vuorimaa, P.: Toward Unified Web Application Development. IEEE IT Professional 13(5), 30–36 (2011)
4. Laine, M., Shestakov, D., Vuorimaa, P.: Extending XForms with Server-Side Functionality. In: 27th ACM Symposium on Applied Computing (SAC 2012), pp. 688–695. ACM, New York (2012)
5. Schmitz, P.: The SMIL 2.0 Timing and Synchronization Model — Using Time in Documents. Technical Report MSR-TR-2001-01, Microsoft Research (2001)