MVC web framework based on eXist application server and XRX architecture ♣

© Yuri Gapanyuk Egor Lakomkin Sergey Ionkin Martin Davtyan

associate professor **student** **student** **student**
gapyu@yandex.ru **egor.lakomkin@gmail.com** **sergey.ionkin@gmail.com** **martin.davtyan@gmail.com**

Bauman Moscow State Technical University

Informatics and Control Systems Department

Abstract

This industrial paper discusses the creation of web framework based on XML technologies. In addition to using eXist application server and XRX architecture, lightweight XML-based data model is designed and XQuery generator for prototyping application infrastructure is developed.

1 Introduction

Nowadays XML-based information systems become widespread. XRX architecture allows creating web application very easily using XQuery and XForms standards.

Our work covers the creation of MVC web framework based on native XML database and XRX architecture.

We have used:

1. XForms – REST – XQuery (XRX) web application architecture.
2. eXist application server that has embedded native XML database.
3. XSLTForms XForms-processor.

We have developed:

1. Framework infrastructure.
2. Simple XML-based data model for storing application data.
3. XQuery generator for prototyping application infrastructure.

We named our framework "iu5 web framework" ("iu5" is a brief name of our subdepartment).

2 Related Work

There are dozens of MVC web framework exist for many programming languages. Probably the most known example is Ruby on Rails [6].

♣ **Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, Moscow, Russia, 2011**

Any MVC web framework allows rapid development of web application infrastructure.

But most of these frameworks are based on classical tree tier web application architecture: database tier (usually relational), web server tier (object-oriented or procedural programming language) and web browser tier (HTML, CSS, JavaScript).

3 Why new XRX-based framework

The weak point of classical tree tier web application architecture is data transfer between tiers because different tiers uses different data formats and different programming language type systems.

XML community invented XRX web application architecture to solve this issue. XRX architecture is described in details in "Why XRX" section of this paper.

XRX approach is very convenient for web developer. And from some point of view XRX approach by itself is very close to be a web framework.

But some features may be added to standard XRX approach e.g. XQuery-scripts code generation, explicit model definition.

Our goal is to develop MVC web framework to do XRX approach more convenient for developers.

4 Why eXist

eXist [3] was created in 2001 as Native XML Database (NXD). eXist could store XML documents in database collections and query XML with XQuery in the first releases.

Now eXist's developer team positions it as a Web Application Server with embedded XML Database.

eXist allows users to use XQuery for creating server pages which is similar to JSP or ASP.NET, therefore it is very easy to create web application with XQuery and store application data in XML format.

Database can store not only XML documents, but binary files too (e.g. images, CSS-files and etc.) which

means eXist has "built in" CMS (Content Management System).

XQuery-scripts also may be stored in database and eXist allows to run them.

Thereby all components of web application may be stored in database and none of them in file system. Or application data may be stored in database and scripts (and other files) – in the file system.

In view of the aforesaid eXist is a very comfortable platform for creating XQuery-based web applications.

5 Why XRX

XRX stands for XForms – REST – XQuery [2]. XRX is example of "zero-translation architecture". It means that there is no translation (data format conversion) between server and client data format. XML is used both on server and client sides and for data transfer between server and client. Thus XRX architecture is simple and friendly for developer.

5.1 Generalized architecture of XRX Web application

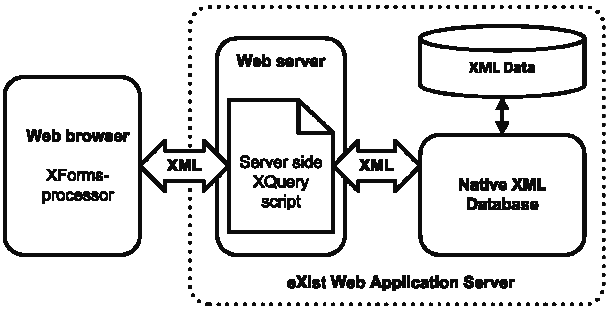

Figure 1: Generalized architecture of XRX Web application.

Figure 1 shows tree tier architecture of XRX Web application.

Presentation tier is XForms-processor. XForms-processor receives XForms-document from Web-server, converts XForms-document to HTML+CSS+Javascript, validates user input, makes XML-fragment from inputted values and sends result XML-fragment back to Web-server.

Application tier is XQuery script that runs on Web-server and communicates with XForms-processor and Native XML Database.

Data tier is a Native XML Database that stores data in XML format.

Data transfer between Application tier and Presentation tier is XML, because XQuery script sends to XForms-processor XForms-documents (that's which is XML), default XML data and receives inputted XML data.

Data transfer between Application tier and Data tier is XML too, because Native XML Database sends and receives data in XML format.

On figure 1 eXist Web Application Server combines functions of a web server, server side XQuery script interpreter and Native XML Database (application tier + data tier).

5.2. Detailed architecture of XRX Web application

Let's examine detailed architecture of XRX Web application on figure 2.

There are tree variants of detailed architecture.

Variant I is example of "shallow XRX" architecture. This is the case where XRX stack is simplified, for example traditional HTML forms are used instead of XForms.

Variants II and III are examples of "deep XRX" architecture where full XRX stack is used.

Variant II is the case when XForms-processor is a client solution. The most popular solutions are XSLTForms [8] and Mozilla XForms [4].

XSLTForms based on XSLT transformation. XSLT transformation may be processed in browser or in server side XQuery-script if necessary. This XSLT transforms XForms-document into HTML + CSS + JavaScript. Resulting HTML-document provides user input, composes XML-fragment from inputted values and sends resulting XML-fragment back to Web-server.

XSLTForms technology is a very flexible solution and it's integrated into eXist.

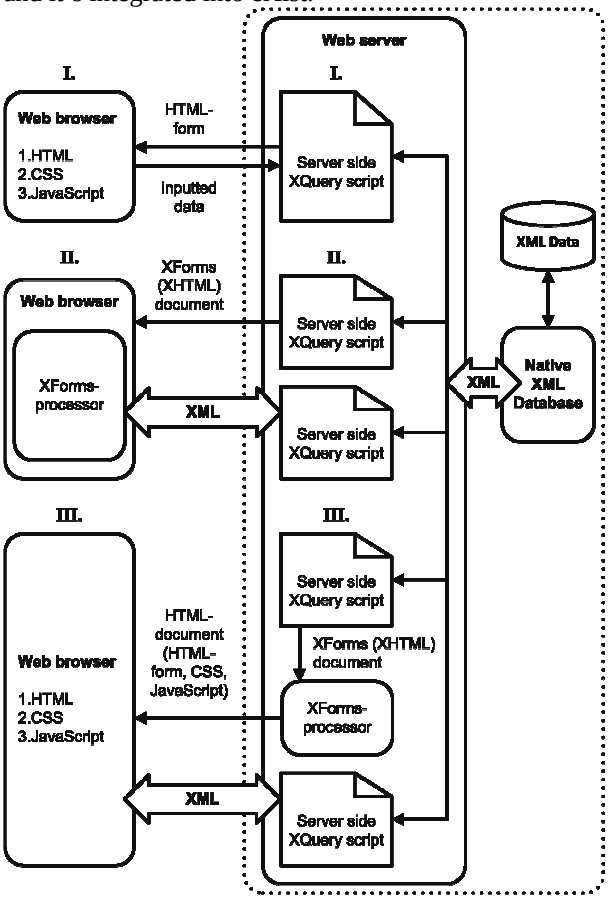

Figure 2: Detailed architecture of XRX Web application.

Mozilla XForms is a plugin for Firefox and other Mozilla products. To our regret it is not as mature as XSLTForms.

Because in variant II XForms-processor is a client-side solution it is very easy to create XForms-documents dynamically and transfer them to client side XForms-processor. This is a great advantage of variant II.

Variant III is the case when XForms-processor is a web server application solution.

The most popular solutions are Orbeon Forms [5] and betterFORM [1].

Orbeon Forms, being a Java web application, is the most advanced XForms-processor. It is a complete application development framework with embedded eXist Native XML Database. But this solution is too complicated.

betterFORM is a successor of Chiba XForms-processor. It is also a Java web application. betterFORM is good solution, but in current version it's difficult to work with dynamically created XForms-documents.

Due to these reasons we choose variant II of application architecture. And we choose XSLTForms XForms-processor because of its flexibility and simplicity to work with dynamically created XForms-documents.

6 iu5 web framework infrastructure

6.1 File system framework infrastructure

Framework uses collections in eXist database for storing data, and file system for storing XQuery-scripts.

For storing XQuery-scripts and other files in file system we use infrastructure shown at figure 3 (folders are shown with border, files are not).

Application folder contains "styles" subfolder for CSS styles, "script" subfolder for JavaScript sources, "modules" subfolder for our "data modules" and "lib" subfolder for additional XQuery scripts and modules.

XQuery script is a file with ".xql" extension which can be executed via HTTP request.

XQuery module is a file with ".xqm" extension. It cannot be executed via HTTP request. It can only be included in ".xql" file.

There is only one "executable" XQuery-script (controller) in application folder, and each module also has one "executable" controller. Other XQuery files are non-executable.

We have minimized number of executable files for web development convenience and for security reasons.

6.2 Data module definition

The main concepts of our framework is "module" (or "data module") and data module element.

The informal definition of data module – it is something like relational table or class in object-oriented language. Data module element – it is something like relational table data row or class object.

The formal definition of data module – it is a combination of XML data (data module elements) stored in database, and files "model.xqm", "view.xqm" and "control.xql".

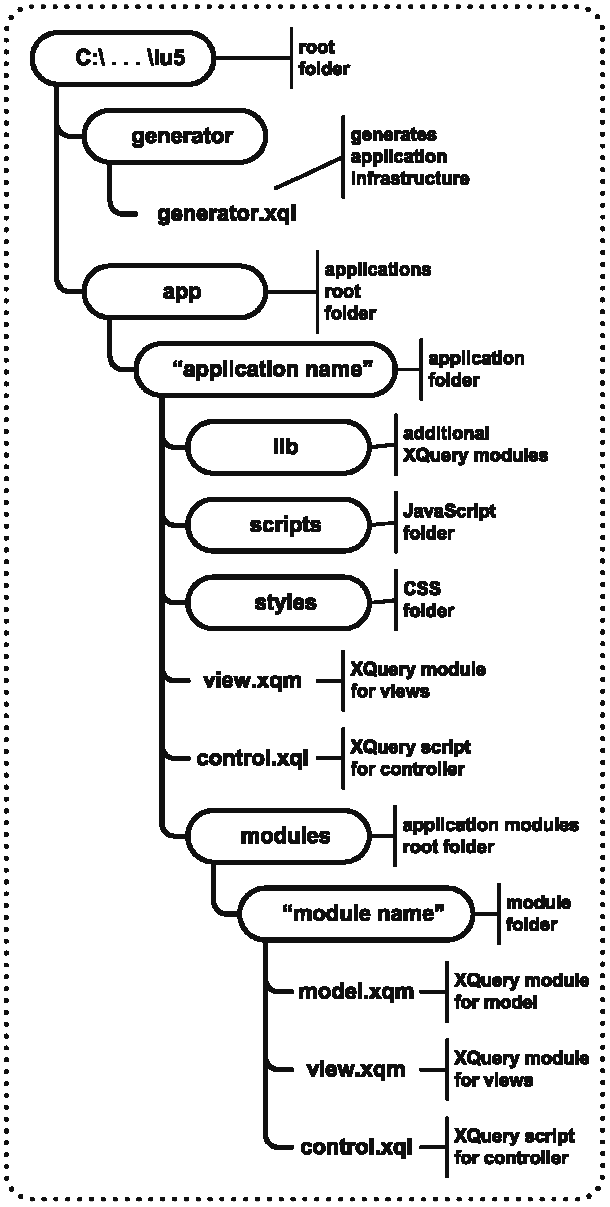

Figure 3: File system framework infrastructure.

Please do not confuse our "data module" with XQuery-module file, which is XQuery library file.

For web development clarity we implemented "classical" MVC pattern for data modules. Each module consists of model, views and controller.

6.3 Data module model

Model consists of XML data stored in database, and file "model.xqm" which contains functions for working with module data.

These are functions for selecting, inserting, updating, deleting and etc. Functions are called from controller.

6.4 Data module view

File "view.xqm" for module contains functions that controller calls for generating views. In current version

we implemented two views for data module: list and edit.

List view generates table of data module elements with buttons for editing and deleting. There is also a button for adding new data module element.

Edit view generates XForms-form for editing one data module element.

File "view.xqm" for application contains only one view, than generate list of hyperlinks for application modules.

6.5 Data module controller

File "control.xql" is controller XQuery-script.

Controller may generate tree kinds of documents:

1. Plain HTML or XHTML documents.
2. XHTML documents with embedded XForms-forms.
3. XML-fragments with default values for XForms-forms.

Controller can receive two kinds of parameters:

1. HTTP POST parameters from HTML-forms.
2. User inputted XML-fragments from XForms-forms.

Depending on parameters controller may perform the following:

1. Call "model.xqm" functions for selecting, inserting, updating, deleting XML data.
2. Call "view.xqm" functions for generating views.
3. Redirect to other modules controllers or to application controller.

For security reasons we use only HTTP POST method. Therefore we do not use any URL rewriting technique, because it requires HTTP GET method.

New versions of eXist has built in "URL Rewriting and MVC Framework" feature. This framework utilizes "controller.xql" controller script and work with URL rewriting. It allows using all kind of models and views.

At first we have decided to use this framework and extend it with our model and views. But later we understood that URL rewriting is not a useful feature for our framework.

Therefore we use controller script name "control.xql" instead of "controller.xql" in order not to conflict with built in eXist feature.

6.6 Database framework infrastructure

For storing XML data in database we use infrastructure shown at figure 4 (collections are shown with border, files are not).

Framework use separate collections for each application and for each application module.

Data module elements stores as XML files in application module collections.

XML file name is data module element unique ID, generated by eXist.

Application module collection is analogue for relational table and XML file is analogue for relational table data row.

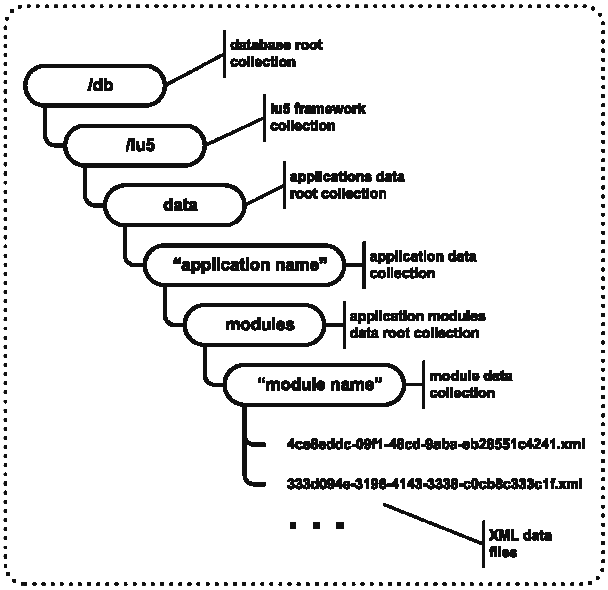

Figure 4: Database framework infrastructure.

7 iu5 web framework data model

7.1 Why new data model

We consider our framework for not very complicated data-centric information systems.

According to our goal we consider following data models for our framework:

1. Relational-like model.
2. Postrelational-like model.
3. XML model based on XML Schema.

For "Relational-like model" we consider XML structure that's have something like "tables", "records", "fields" and "relations" described by XML tags.

But this "classical" model has many restrictions. For example, if we want to model repeated groups of fields we have to create new table.

From our point of view "Postrelational-like model" is much more suitable for data-centric information systems. We consider "Postrelational-like model" as "Relational-like model" with 1NF violation. Therefore we can use groups and repeated groups of "fields" in one "record".

On the other hand XML model based on XML Schema is too "free-form" for our framework, because our framework is data-oriented, not document-oriented.

No doubts, with XML Schema we can describe very complicated combinations of "sequences" and "choices" but is not easy to generate XForms-form based on XML Schema. Some experimental converters from XML Schema to XForms already developed (e.g. xsdTransformer [7]) but it is risky to use them for production.

Therefore "Postrelational-like model" is a happy medium between restrictive "Relational-like model" which is inconvenient for developer and "XML model based on XML Schema" which is difficult for generating correct XForms-forms.

7.2 Data model overview

Our “Postrelational-like” data model is very simple. It contains only three XML tags:

1. Simple value. Example:
   <v n="name">value</v>
2. Reference to value in current module or in other module. Example:
   <r n="name" module="module name">value</n>
   Attribute “module” not used in case of current module reference.

   XPath expressions for selecting values not pointed in “r” tag. These expressions pointed in application generator data and then generator “built in” them into generated XForms-form.
3. Group element. Example:
   <g id="auto generated id" n="group name">
   Nested elements “g”, “r” or “v”.
   </g>

   In XML file root element “g” corresponds to data module element. It may have nested elements “g”, “r” or “v”.

   There are three types of groups:

1. Simple group. Used for encapsulation of “r” or “v” elements.
2. Repeated group. Sequence of subgroups for modelling list of values. Analogue of “sequence” XML Schema construction.
3. Variant repeated group. Sequence of various subgroups for modelling list of heterogeneous values. Analogue of “sequence” of “choice” XML Schema construction.

   Group type pointed in application generator data and then generator consider it when generating XForms-form.

8 iu5 web framework application generation

Application generator script generates XQuery web application according to XML declaration.

There is an interesting fact that we use our data model both for application data and for application declaration.

Let’s examine detailed example of application generation.

8.1 Application declaration

We define sample application as test constructor for simple testing system. The only type of question in this system is question with multiple choices.

We’ll declare data model for tests and questions and framework will generate forms for list view and edit.

Of course our framework will help us only with test constructor. The questioning module should be written by hands with “traditional” XRX.

Text of application declaration is shown below:

```
<g id="" n="application">
 <v n="app_name">test</v>
 <v n="app_caption">Test constructor</v>
 <g id="" n="modules">
  <g id="" n="module">
   <v n="module_name">question</v>
   <v n="module_list_caption">List of
questions</v>
   <v n="module_edit_caption">Question
edit</v>
   <g id="" n="list_data">
    <g id="" n="list_fields">
     <v n="list_field">question_text</v>
    </g>
    <v n="order_field">question_text</v>
   </g>
   <g id="" n="fields">
    <g id="" n="group">
     <v
n="group_name">question_main_group</v>
     <v n="group_caption">Question
params</v>
     <g id="" n="fields">
      <g id="" n="field">
       <v n="name">question_text</v>
       <v n="caption">Text</v>
       <v n="type">xforms:string</v>
       <v n="required">true()</v>
       <v n="alert">This field is
required</v>
       <g id="" n="string_details">
        <v n="width">333</v>
        <v n="height">25</v>
       </g>
      </g>
      <g id="" n="field">
       <v n="name">question_time</v>
       <v n="caption">Time limit</v>
       <v n="default">00:00:00</v>
       <v n="type">xforms:time</v>
      </g>
     </g>
    </g>
    <g id="" n="repeat_group">
     <v n="group_name">answers</v>
     <v n="group_caption">Answers
list</v>
     <v n="group_formtype">list_num</v>
     <g id="" n="repeat_items_group">
      <g id="" n="repeat_item_group">
       <v n="group_name">answer</v>
       <v n="group_caption">Answer</v>
       <g id="" n="fields">
        <g id="" n="field">
         <v n="name">answer_text</v>
         <v n="caption">Answer text</v>
         <v n="type">xforms:string</v>
         <v n="required">true()</v>
         <v n="alert">This field is
required</v>
         <g id="" n="string_details">
          <v n="width">333</v>
          <v n="height">25</v>
         </g>
        </g>
        <g id="" n="field">
         <v n="name">answer_right</v>
         <v n="caption">Right answer</v>
         <v n="type">xforms:boolean</v>
         <v n="default">false</v>
         <v n="alert">At least one answer
should be marked as right</v>
```

```
        <v n="constraint">
(count(//*[@n='answer_right' and boolean-
from-string(.)=true()]) &gt; 0) </v>
        </g>
       </g>
      </g>
     </g>
    </g>
   </g>
  </g>

  <g id="" n="module">
   <v n="module_name">test</v>
   <v n="module_list_caption">List of
tests</v>
   <v n="module_edit_caption">Test
edit</v>
   <g id="" n="list_data">
    <g id="" n="list_fields">
     <v n="list_field">test_name</v>
    </g>
    <v n="order_field">test_name</v>
   </g>
   <g id="" n="fields">
    <g id="" n="group">
     <v n="group_name">
test_main_group</v>
     <v n="group_caption">Test params</v>
     <g id="" n="fields">
      <g id="" n="field">
       <v n="name">test_name</v>
       <v n="caption">Test name</v>
       <v n="type">xforms:string</v>
       <v n="required">true()</v>
       <v n="alert">This field is
required</v>
       <g id="" n="string_details">
        <v n="width">333</v>
        <v n="height">25</v>
       </g>
      </g>
     </g>
    </g>
    <g id="1" n="repeat_group">
     <v n="group_name">test_questions</v>
     <v n="group_caption">Test
questions</v>
     <v n="group_formtype">list_num</v>
     <g id="" n="repeat_items_group">
      <g id="" n="repeat_item_group">
       <v
n="group_name">test_question</v>
       <v n="group_caption">Question</v>
       <g id="" n="fields">
        <g id="" n="ref">
         <v
n="name">test_question_ref</v>
         <v n="required">true()</v>
         <v n="alert">This field is
required</v>
         <v
n="ref_formtype">select_list</v>
         <v n="distinct">true()</v>
         <v n="pathgroup">question</v>
         <v
n="pathlabel">question_text</v>
         <v n="module">question</v>
         <v n="appearance">minimal</v>
         <g id="" n="string_details">
          <v n="width">333</v>
          <v n="height">100</v>
         </g>
        </g>
       </g>
      </g>
     </g>
    </g>
   </g>
  </g>
 </g>
</g>
```

8.2 Example of generated XML data model

Example of generated question XML data file is shown below:

```
<g id="12956faf-9d4e-4b79-bc5c-
9665302bc58e_2011_5_4_4_51_18F82"
n="question">
 <g id="question_main_group_1_2011-05-
04T00:51:25Z" n="question_main_group">
  <v n="question_text">Select XML
technologies</v>
  <v n="question_time">00:00:00</v>
 </g>
 <g id="answers_1_2011-05-04T00:51:25Z"
n="answers">
  <g id="answer_1_2011-05-04T00:51:25Z"
n="answer">
   <v n="answer_text">XForms</v>
   <v n="answer_right">true</v>
  </g>
  <g id="answer_2_2011-05-04T00:51:26Z"
n="answer">
   <v n="answer_text">HTML</v>
   <v n="answer_right">false</v>
  </g>
  <g id="answer_3_2011-05-04T00:51:31Z"
n="answer">
   <v n="answer_text">XQuery</v>
   <v n="answer_right">true</v>
  </g>
 </g>
</g>
```

Example of generated test XML data file is shown below:

```
<g id="f8cac716-d965-4590-a189-
1de0139c6706_2011_5_4_5_2_22F76"
n="test">
 <g id="test_main_group_1_2011-05-
04T01:02:32Z" n="test_main_group">
  <v n="test_name">XML test</v>
 </g>
 <g id="test_questions_1_2011-05-
04T01:02:32Z" n="test_questions">
  <g id="test_question_1_2011-05-
04T01:02:32Z" n="test_question">
   <r n="test_question_ref"
module="question">12956faf-9d4e-4b79-
bc5c-9665302bc58e_2011_5_4_4_51_18F82</r>
  </g>
 </g>
</g>
```

8.3 Examples of XForms-forms

Examples of generated XForms-forms are shown below.

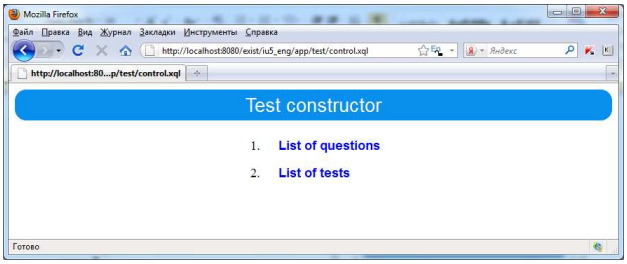

Figure 5: List of application modules form.

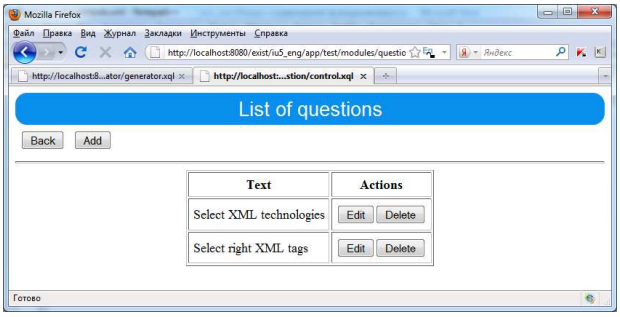

Figure 6: List of questions form.

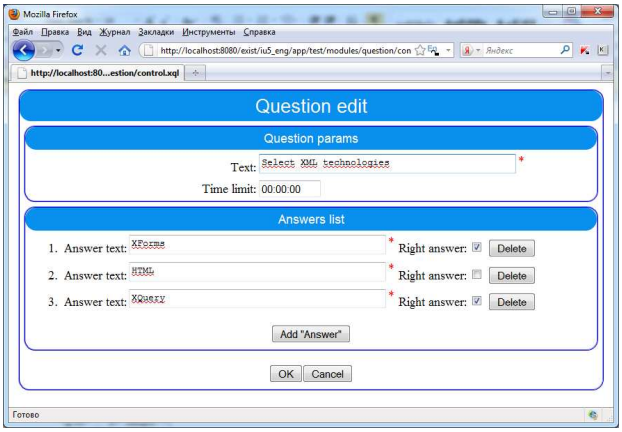

Figure 7: Edit question form.

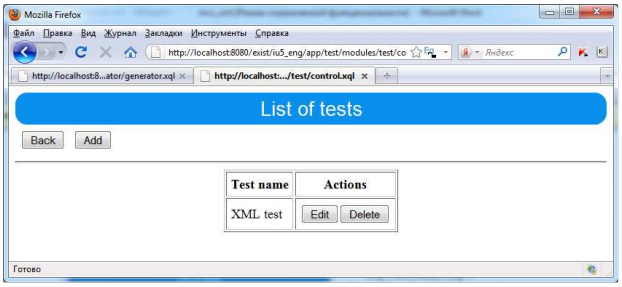

Figure 8: List of tests form.

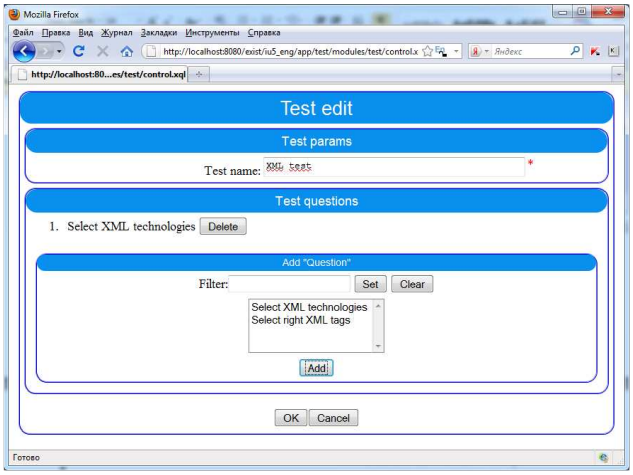

Figure 9: Edit test form.

9 Conclusion and future work

In this paper we introduced the first version of our “iu5 web framework” based on XRX architecture and eXist web application server.

We are going to improve our framework in the following ways:

1. Add embedded security, which is the most critical feature.
2. Improve views generation and user interface.

References

[1] betterFORM Web site, 2011. http://www.betterform.de

[2] Cagle Kurt. Metaphorical Web and XRX, 2008. http://broadcast.oreilly.com/2008/09/metaphorical-web-and-xrx.html

[3] eXist Web site, 2011, http://exist-db.org

[4] Mozilla XForms Web site, 2011. http://www.mozilla.org/projects/xforms

[5] Orbeon Forms Web site, 2011. http://www.orbeon.com

[6] Ruby on Rails Web site, 2011. http://rubyonrails.org

[7] xsdTransformer Web site, 2011. http://xsdtrans.sourceforge.net

[8] XSLTForms Web site, 2011. http://www.agencexml.com/xsltforms