XIDE Component Description

Document Name	XIDE Component Description
Editors	Evgenia Samochadina, samochadina@gmail.com
Version	Version 1.0
Date of Last Change	30.06.2010

Table of contents

1 Introduction

1.1 Purpose of the system

XIDE is a web-based visual editor for developing XFormsDB-based web applications. It is designed to help users to create and manage web applications, which solve their personal tasks. It provides different features to facilitate development process at all stages, e.g. wizard-based application creation, visual page content creation, management of published applications.

XIDE is focused on *non-professional users.* It allows to create highly interactive application without writing single line of code or having technical knowledge of XForms or XFormsDB at all. However, more *professional users* are also taking into account. XIDE provides functionality to improve the process of application development in comparison with doing the development in plain text editor or IDE.

1.2 Objectives of this document

This document is designed for developers, who are going to continue XIDE development. It covers process of XIDE development, XIDE main components and their relationship,

More information about XIDE implementation can be found in the comments in the XIDE source code.

1.3 Required Background knowledge

In order to be able to continue XIDE development, one should have the following background:

- Java
- Google Web Toolkit (GWT)
- HTML and CSS
- JavaScript

1.4 Other resources

There are several resources where more information about XIDE can be found:

- Evgenia Samochadina's Msc Thesis "XIDE - a visual component-based web application editor" http://xformsdb-ide.googlecode.com/files/Msc%20Thesis%20draft.docx contains detailed description of process of XIDE design, implementation and evaluation
- XIDE open source project page on http://code.google.com/p/xformsdb-ide/ provides source codes and technical details about XIDE installation and deployment
- XIDE welcome page http://testbed.tml.hut.fi/xide/ contains demo videos and information about demo applications

Questions and comments are highly welcome by email samochadina@gmail.com

1.5 Document overview

This document explains briefly how it was developed (including development environment and development process description); what are main modules and how do they communicate with each other; how does XIDE works on production server.

For information about installation, setting up development environment and deployment please refer to corresponding documents.

2 System description

XIDE is client-server web-based application.

Client-side is responsible for UI representation and managing user actions. It sends requests to the server in response to user activity and processes server response.

Server-side implementation consists of three servlets, which are responsible for the communication with the client, and several supporting modules. Main server responsibilities are communication with the database, managing file structure, publishing the application etc.

2.1 Module diagram

Module diagram shows logical structure of the system and dependencies between modules. *Uses* relation is used in this diagram. Elements of the view are modules, which are created based on implementation packages.

For reader's convenience, the module diagram is separated into two parts: client- and server-side. Client-side modules and their relations are shown in Figure 1 and described in Table 1. Server-side modules and their relations are shown in Figure 2 and described in Table 2.

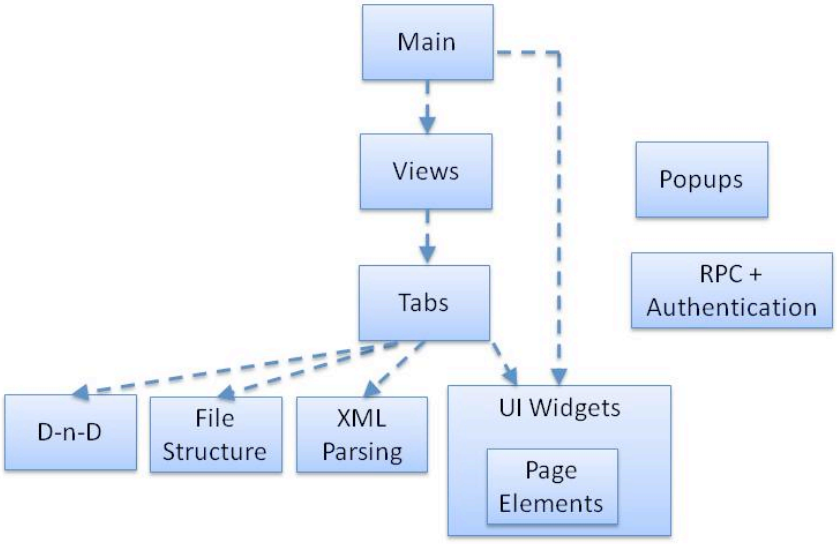

Figure 1 XIDE client-side modules and their relations

Table 1 XIDE client-side modules description

Module name	Description
Main	Main class takes care about XIDE client-side initialization process.

	After that, it manages different application views, switching to corresponding view according to both XIDE user actions and browser back and forward events. It is also responsible for corresponding header links management. Finally, it handles events, which are sent by different elements and needs to be propagated.
Views	Contains set of Views classes. Each of them is responsible for one view (Welcome page, Application List, Application, Page). Common functionality is event propagation management. Each view initiates the tabs it consists of and displays them on the screen accordingly.
Tabs	Tabs package contains set of different Tabs. Common functionality is possibility to update the displayed information according to received event. Each tab is responsible for rendering the information it contains and update it according to the events received.
UI Widgets	UI widgets package contains different custom UI objects to be used on the Tabs (e.g. Panels, styled buttons etc.).
Page Elements	Page elements are Component, Container and Web Page. Each element is UI representation of corresponding Template Language abstraction and takes care about both logic and graphical representation of the object.
D-n-d	D-n-D package contains classes responsible for drag-n-drop process. It is utilized by several Tabs, which has drag-n-drop functionality.
File Structure	File structure's main responsibility is to provide abstractions, which represent physical files and folders in XIDE. It is utilized by File Structure Tab, which shows physical file structure of the object at the server (e.g. application or component).
XML parsing	XML parsing takes care about parsing and rendering XFormsDB on the client-side. More information can be found in the subsection ***.
Popups	Popups package contains custom popup windows with different styles and purposes. There are general popups, e.g. Error Popup or Notification popup, which are widely used in the system. There are also custom forms and wizards; they are used in special cases, e.g. New Application Wizard or Tag Search popup.
RPC + Authentication	RPC is responsible for communication with the server by means of Remote Procedure Call; authentication process is integrated with RPC.

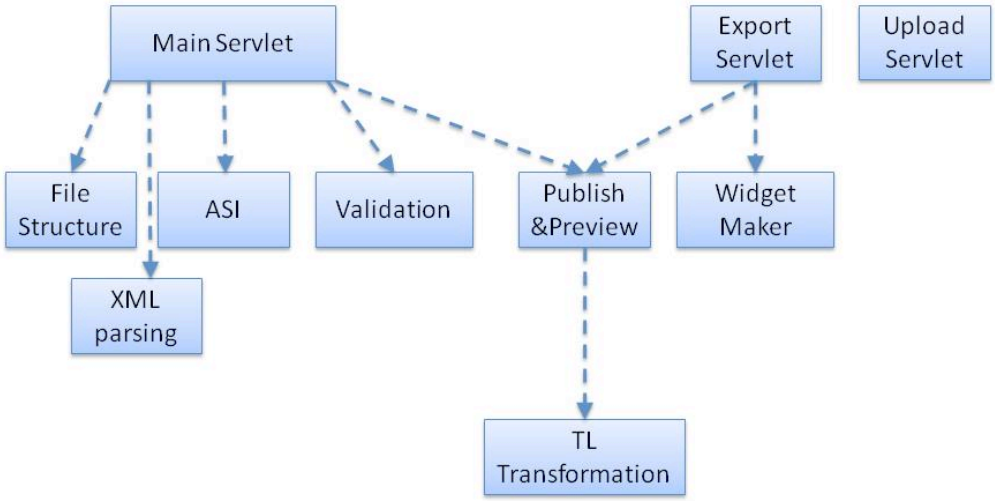

Figure 2 XIDE server-side modules and their relations

Table 2 XIDE server-side modules description

Module name	Description
Main Servlet	Main Servlet is responsible for communication with client based on RPC calls. It process requests to the database and forwards requests to other modules
File Structure	This module is responsible for creating and parsing file structure on the server. It is used when new application is created or when page is requested from the client.
ASI	Module takes care about communication with ASI server to perform authentication-related tasks
Validation	This module validates pages and components syntax. It is used when page or component is saved.
Publish & Preview	Publish & Preview module is responsible for application publishing and preview. It manages transformation, copies necessary files, initializes Exist database and deploys the application.
TL transformation	This module takes care about transformation from internal Template Language into XFormsDB. This includes substitution of parameters and component's calls with valuable source code.
Export Servlet	This Servlet handles client requests to download files from server. It is used to export the application as a widget.

Widget Maker	This module is responsible for creating archive with the published application. This archive can be used to deploy the application on the external server.
Upload Servlet	This Servlet processes client requests to save a file

2.2 Client-server diagram

Client-server diagram illustrates how client and server communicate in the runtime. Elements in this view are client and server components and protocol connectors.

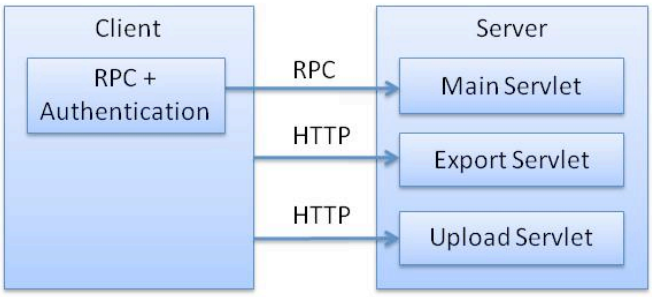

Figure 3 Client-server diagram

Component name	Component description
Client	Client part of application running on user's browser on user's computer.
Server	Server part of the application running on the external server. Three servlets are responsible for communication with the client.
RPC + Authentication	This module is responsible for communication with the server by means of RPC. All other modules use this module in order to send the RPC request to server and receive a response.
Main Servlet	This Servlet manages all RPC requests, received from the client.
Export Servlet	This Servlet process client HTTP requests to download a file.

Upload Servlet	This Servlet process client HTTP POST request to save the file.
RPC	RPC is a communication technology used to remote invocation. In XIDE it is used to implement asynchronous AJAX request to the server to perform some server-side action. Often it includes communication with the database and/or managing file system.
HTTP	A standard HTTP request

Table 3 Description of client-server diagram components

2.3 Database schema

XIDE uses relational database to store internal data about applications and templates. Generally, all information about applications and templates are stored on the server in the format of files. Database contains only major information (titles, descriptions, dates, etc.), which is used for search purpose. Information in the database is retrieved and updated by the server part of the XIDE. Database schema is displayed on Figure 4 and table descriptions can be found in Table 4.

For more information about tables, their columns and content, please refer to the database installation scripts on the XIDE project page.

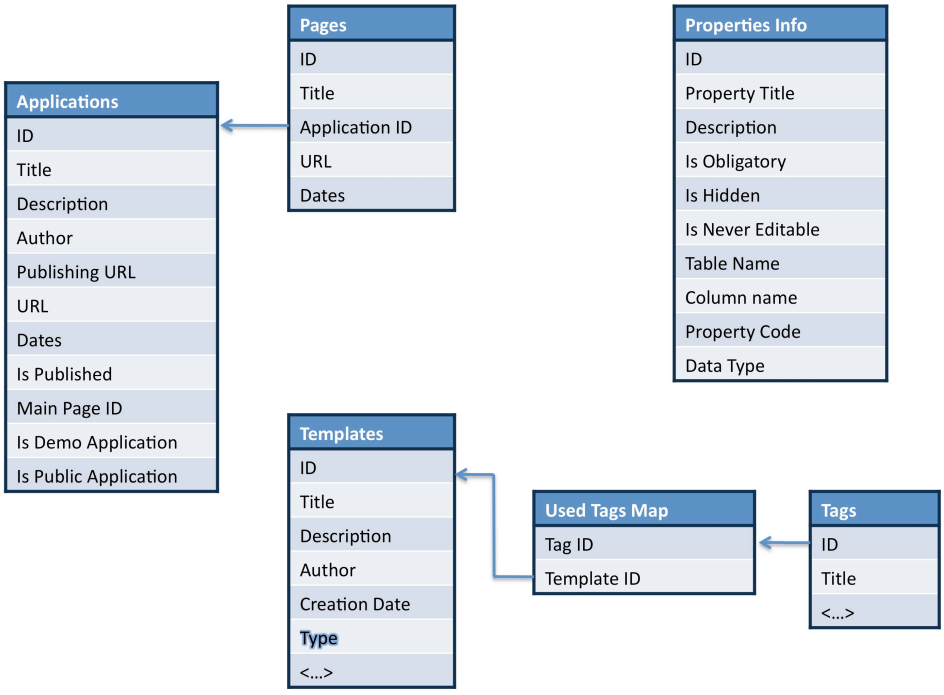

Figure 4 XIDE database architecture

Table name	Table description
Applications	This table stores information about applications created by users
Pages	This table stores information about pages created by users. Each page belongs to one single application.
Templates	Templates table stores information about elements, that can be reused (component templates and pages and application samples) In this table each entry is called template; each template is a reference to a reusable item and contains some information about it, used for search purpose. When the user uses template, all information about the template is parsed from the file system.
Tags	Tags table stores information about different XIDE tags. When the user creates tag, it is added to the table. Each template element (component templates and pages and application samples) can have one or more tag assigned.
Used_Tags_Map	This table contains information about tag assignments. It stores many-to-many relationship between tags and templates tables
Properties_Info	This table stores information about properties that can be parsed from all other tables.

Table 4 Description of tables, used in XIDE

2.3.1 Concept of property in XIDE

In XIDE information about each object is represented as a set of properties. Each property contains the value and information about it: how is it called (title), what is it for (description), can it be edited by the user, etc. For example, there is a property to store URL of the application

value: *www.xide.com/application.xformsdb*

title: *Applciation URL*

description: *URL where the application is published and can be accessed externally*

can be edited by user: *no*

is hidden: *no*

There are many different property types; objects of the same type have the same set of properties. Generally, each column in the table, where object information is stored, should be parsed into single property object. Except the value, two properties of the same type have the same descriptive information (title,

description, different flags). For each property type there is an entry in *Properties_Info* table.

When the object is parsed, corresponding properties are created according to the information from *Properties_Info* table, and the values from the object information are stored into those properties.

2.4 Deployment diagram

Deployment diagram shows how the system distributed among the network during deployment. It describes how the XIDE is deployed on production server and its environment. Deployment schema is displayed on Figure 5 and descriptions of components, displayed on the schema, can be found in Table 5

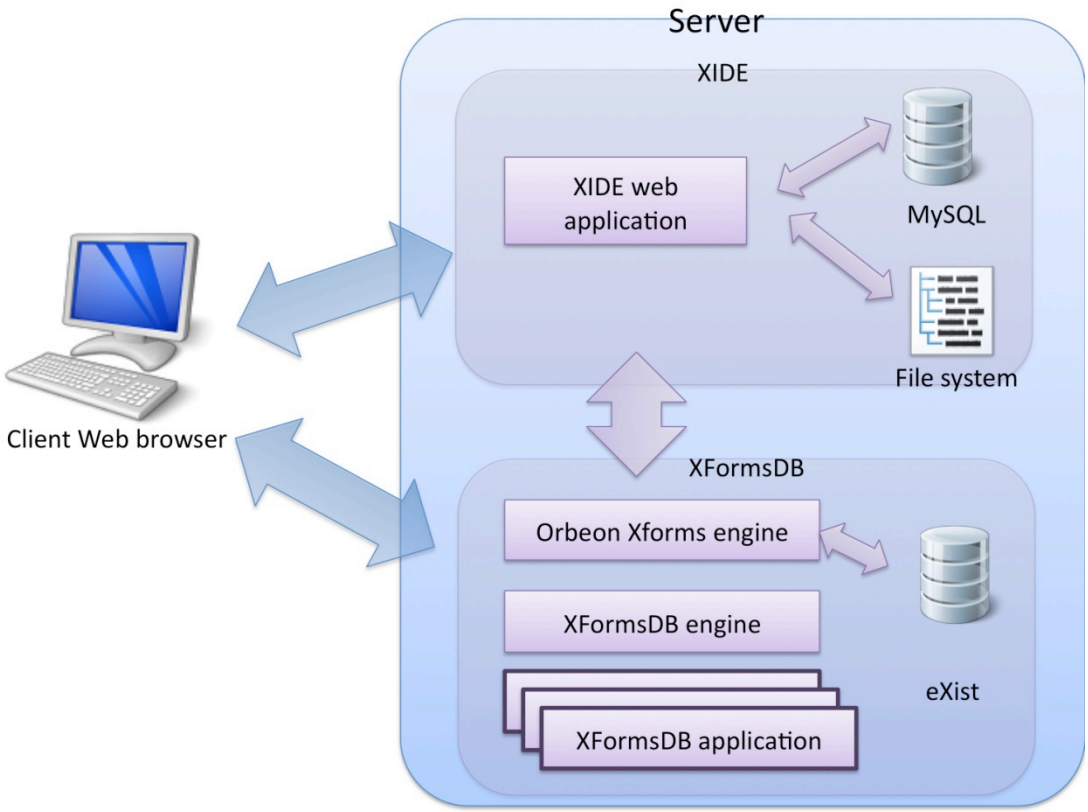

Figure 5 XIDE Deployment view

Component name	Component description
Client Web browser	User accesses the XIDE from the browser on user's computer. Also user can access the XFormsDB application directly.
Server	Apache Tomcat web server supports both XIDE and XFormsDB engine. However, it is not obligatory to have both applications running on the same server.

XIDE	XIDE is deployed as a Web Application archive (WAR). [Cite war] It contains web pages, related pictures, JavaScript scripts, and servlets to process server-side functionality. It communicates with XIDE database (MySQL) and XIDE files (File system). While XIDE processes user action, it may require communicating with XFormsDB engine. This happens when user requests to view the page, written on Template Language (e.g. preview application).
MySQL	MySQL relational database contains information about components, users, web pages and applications. It also supports tags; currently tag can be assigned only to component.
Files system	XIDE physical files include components' and application sources and published and previewed applications. They are organized and stored on the server.
XFormsDB	This part represents XFormsDB related part of the system.
XFormsDB engine	XFormsDB engine is a web application deployed on the server. It takes care about processing XFormsDB applications. (Laine, 2010) Its main responsibilities are processing of XFormsDB into XForms transformation, communication with Exist database and processing requests to display XForms applications.
XFormsDB applications	XFormsDB applications are deployed to the server and can be accessed both by XIDE and user's browser. In order to be displayed in the browser, each application should be processed by XFormsDB engine. (Laine, 2010)
eXist	eXist XML database used by XFormsDB applications to store the data. XQuery and XPath can be used to manipulate with the data.

Table 5 XIDE Deployment diagram

3 Development process

3.1 Main technologies for development

Google Web Toolkit was selected to be main technology for both client- and server-side. It is free, open source framework for developing high interactive client-server web applications.

Basic programming language of both client-side and server-side code is Java; on the client-side it is combined with HTML, CSS and JavaScript. Client-side Java code is automatically transformed into JavaScript during the compilation phase.

The development has started with GWT version 1.4 and Java version 1.5, later it was upgraded up to GWT 1.7 and Java 1.6. In this project GWT combined with pure Java, XHTML, Javascript, CSS, and XSLT was used to implement most of client-side and server-side functionality.

3.2 Development process

XIDE development was organized so that visual results appeared as fast as possible. There were two reasons for doing this. Firstly, the nature of the project required often demonstrations of the results for the research group. Secondly, early prototyping allows start discussion and testing of the interface at the beginning on the development, what is really important in case of user- and usability-focused system.

First prototype of the XIDE demonstrated further system layout only; it had no functionality implemented. It was based on paper-based mockups, which were developed during design phase. Later hardcoded parts were gradually replaced with full-functional ones and new functionality was added.

When level of prototype became sufficient for testing it with users, a first round of usability testing was made. Further development continued based on rapid prototyping principal.

4 Future work

XIDE development process has reached some valuable result and now it is reasonable to offer it to external people to try it out. However, there are several features to be developed:

- Web 2.0 features (comment a component from the database or subscribe to updates for the component)
- Sample pages and applications; pages with different layouts, offered to a user during page creation process
- Component versioning; being able to update component version and manage several versions of component during creation and further management of the page
- CSS problem; currently users should create their CSS files without ability to use UI element name from the main source file (XFormsDB). This problem caused by XForms processor, which changes element names. So user has to know how the element will be called after it will be transformed to HTML in order to be able to assign CSS rule for it. Trying to overcome this problem or provide some features to make it less confusing to users.

- Using components implemented based on other technologies; can be useful to cooperate with other technologies and component databases.