

QSYS

Developer Guide

Sabine Grabner

Version 1.0 beta

Last Change: June 29th 2009

Abstract

This guide should help contributors to get quickly up to speed with the project's code base. It gives an overview of the software architecture and technologies used.

Table of Contents

SOFTWARE ARCHITECTURE	4
DEVELOPMENT WITH ECLIPSE	4
BUILD	4
TECHNOLOGY	5
AUTHENTICATION AND AUTHORIZATION	5
TYPES OF USERS	5
SECURITY VOCABULARY	5
AUTHENTICATION	5
AUTHORIZATION	6
PROPOSED POLICY ENFORCMENT POINT (PEP) IN QSYS-CORE	7
I18N	7
INTERNATIONALIZATION OF CONTENT	7
INTERNATIONALIZATION OF THE APPLICATION	7
QSYS-CONTRIB	8
ANALYSIS	8
4 TYPES OF ANALYSES:	8
PROPOSED REFACTORING	9
CLASS HIERARCHY:	9
VISUALIZATION STYLES	9
EXCEPTION HANDLING	9
API	9
DEV:	9
QSYS-WEB	10
JAVA RESOURCES	10
XSL TRANSFORMATION	10
JAVASCRIPT	11
TABBING	11
RICH TEXT EDITING	11
RECOMMENDED STEPS FOR ADDING NEW FEATURE	11
SUGGESTED FEATURES	11

Software Architecture

QSYS consists of the following sub projects:

qsys-core

The Java API.

qsys-contrib

Contributions to qsys-core.

qsys-web

A Java web client using the API.

q-utils

This project consists of useful and reusable Java methods.

q-struxsl

This project is an extension of the Struts web framework and builds the web framework used for qsys-web. Additionally to Struts' use of JSP pages and tag libraries, q-struxsl allows Action classes to output XML that is rendered to HTML or PDF through XSL templates.

attributeBasedAuthorization

This project supports various authentication mechanisms for Java web applications. It adds the access control flexibility needed when deploying QSYS at larger organizations.

Development with Eclipse

1. Install Eclipse if not present
2. [Install and configure the Sysdeo Tomcat Plugin](#). Also follow installation instructions of `eclipse_home/plugins/com.sysdeo.eclipse.tomcat_3.2.1/readmeDevLoader.html`
3. Repository check-out: check out all projects described above (for CVS access please contact albert.greinoecker@uibk.ac.at)
4. Configuration: see the Administrator Manual
5. Right click qsys-web -> Tomcat Project -> Update context definition
6. Start tomcat (read sysdeo tomcat plugin for instructions)
7. Launch qsys at <http://localhost:8080/qsys>

Build

Due to the lack of an ant build script at this point qsys-contrib/dist/qsys-contrib.jar needs be built first. Then build qsys-web/dist/qsys.war:
cd qsys-web

```
chmod u+x build.sh
./build.sh
```

Technology

qsys-core was developed using JDK 1.5. qsys-web was developed using the Jakarta Struts framework. Version? Data are stored in XML. XSLT or XSL-FO transformation was used for rendering to HTML and PDF respectively. Runs on Java Servlet engine. Tested with Tomcat.

Authentication and Authorization

For QSYS' access control an independently developed authentication middleware is being used that offers several authentication mechanisms. For authorization the middleware suggests using XACML.

Types of users

- Administrator
- Author
- Interviewee

Security vocabulary

qsys-core:

- *org.qsys.security.policy.SubjectAttribute:*
subject attribute names used in the XACML policies
- *org.qsys.security.XACMLAction:*
action IDs used in XACML policies

qsys-web:

- *org.qsys.security.QSYSGroup:*
not in use
- *org.qsys.security.QSYSRole:*
lists all roles a user can possess in QSYS. Currently it lists 'admin', 'groupadmin', and 'interviewee'. The initial naming 'groupadmin' should be changed to 'author' in a future version to reflect the role's real purpose.

Authentication

Authentication is enforced by the qsys-web client. qsys-web can be configured to use either of the following authentication mechanisms:

- QSYS' native authentication
- LDAP (through authentication middleware)
- Shibboleth (through authentication middleware)

Starting points of authentication cycle:

- Administrator: org.qsys.quest.action.process.AdminLoginProcessAction
- Author: org.qsys.quest.action.process.GroupAdminLoginProcessAction
- Interviewee: org.qsys.quest.action.process.AnswerLoginProcessAction

The author's and interviewee's credentials can be validated against either of abovementioned authentication mechanisms. The admin password is stored in qsys.properties.

Logic of LDAP authentication in qsys-web

1. After user name and password have been entered by the user, the following session variables are being set by the authentication enforcement points:
 - LDAPAuthentication.args.user.name()
 - LDAPAuthentication.args.pw.name()
 - AuthenticationServlet.POST_AUTHENTICATION_URL_TO_GO_TO
2. User is being forwarded to the LDAPAuthentication servlet
3. Username and password are used by the servlet, which then sends the user to the post-authentication URL. If the authentication succeeded an at.fhv.security.User object is being added to the session variables. If any exception occurred the reason is being added to the session variables and further processed in the next step.
4. The URL to go to after authentication depends on whom to authenticate:
 - PostAuthoringAuthentication (GroupAdminLogin)
 - PostIntervieweeAuthentication (AnswerLogin)

Both servlets inherit from AuthenticationServlet and take care of QSYS specific settings after login.

5. Upon failure the user gets sent back to the login interface. Upon success the user gets added to a blacklist to prevent multiple participations, then they get anonymized for privacy reasons and finally redirected to the protected resource requested.

Authorization

XACML technology defines a policy schema and supports evaluating access requests against such policies.

An XACML access request consist of a subject, a resource and an action:

- *Subject*: An object of type org.qsys.security.User (see vocabulary for allowed attribute names). qsys-web overrides the middleware's User object to restrict group and role assignment to a sole one. This design decision was necessary for backward compatibility of user accounts with QSYS' legacy/native authentication.
- *Resource*: the questionnaire or template ID
- *Action*: any action of the action vocabulary

Proposed policy enforcement point (PEP) in qsys-core

Public methods of the QSYS API should be accessible through classes that inherit from PEP. Methods of such subclasses support specific QsysActions. The constructor of PEP needs a User object, a resource ID and a QsysAction, where the QsysAction is hard coded in the inheriting class. If an XACML request validates positive the object is created and the class' methods can be accessed. Methods that do not require access control are supposed to be static, where possible.

TODO: Create table of QSYS actions and qsys-web icons

I18n

It is important to distinguish between internationalization of the application and internationalization of content.

Internationalization of content

I18n is only supported by the file system storage package. It was decided that methods in the db package consume the locale as a String value as opposed to a Locale object.

Reading localized content:

As long as writing localized content is not supported by qsys-core, it is advised not to read localized content for authoring purposes. However, localized content can be read for interviewee purposes if available.

```
// getting locale in qsys-web:  
String lang = sessq(request).getCurrentLocale().toString();  
String lang = request.getLocale().toString();
```

```
// reading localized document  
QuestionIntervieweeManager qMan =  
    sessq(request).getQuestionIntervieweeManager();  
Document doc = (Document) qMan.getQuestionnaireContent(lang);
```

Internationalization of the application

The applications language choice consists of English and German and can be easily augmented. The language documents are stored in qsys-web/webroot/lang.

qsys-contrib

There are currently two contributions allowing automated analysis of questionnaires and e-mail notifications to the questionnaire author upon participations.

Contributions also use the XML format for persistence of settings. However, the difference to qsys-core is the use of XSD schemas and JAXB binding for (de)serialization.

Analysis

The analysis feature allows downloading PDF documents with analyzed survey results. For closed questions statistical values such as mean and standard deviation can be displayed. Most question types can be displayed in various visualization styles.

Preferences as for the visualization style, legend position, and the like are stored in XML format.

There are four ways of analyzing questionnaires. In the simplest case a single questionnaire is analyzed and the result is put into a PDF document. However, it is also possible to analyze multiple questionnaires at once if they are instantiated from a common template. There are two choices for the output of batch analysis. Either each questionnaire's result is put into its own PDF document and archived together to a zip file, or all questionnaires' responses are accumulated appearing as a single questionnaire's responses and result into a single PDF document just like the simplest case described above. The most complex case allows grouping questionnaires, and then accumulating all questionnaires' responses within each group, which then reflect one questionnaire per group. These questionnaires can be visualized in a comparative fashion.

4 types of analyses:

	What is analyzed	output	Notes
Single	1 questionnaire	PDF	Is like Individual with one questionnaire only.
Individual	multiple questionnaires	Zipped PDFs	
Accumulated	multiple questionnaires	PDF	Is Parallel with one group only
Parallel	multiple questionnaires grouped, each group has one or more questionnaires	PDF	Each group's results are displayed in graphs for direct visual comparison

The analysis type is not stored with other analysis preferences. It is usually picked in the client UI when generating the analysis.

Proposed Refactoring

Back when 'accumulated' was implemented, merging of multiple AbstractAnalysis objects of the same type was not available. For code maintainability the new merge possibility should be used.

Class Hierarchy:

Subclasses of Single have a DataSetFrequencyDistribution object, and subclasses of Matrix have a DataSetFrequencyDistributions object.

Visualization Styles

Text: delimiter has to be set for same answers being counted, hmmm

Exception Handling

When the PDF is written to the servlet output stream, thrown exceptions are not properly handled yet.

API

Parallel analysis: allows putting questionnaire IDs into several Vectors, the vectors are put into a TreeMap with a name for the grouped IDs within a vector as key.

Dev:

JFreeChart API

for compiled, all answers get put into the first object of the treemap, the others are left untouched

if client hands output stream, he has to hand over a zip stream for individual, and a regular os and questID for individual

if client hands no output stream, the compiled version is put to the template dir as analysis_timeAsMillis.zip or individual pdfs are stored in the quests directory,

AnalysisManager: analyzes questionnaire, analyzes all answers, puts them into applicable dataset, compiles all infos into PDF doc

AnalysisConfigurationManager: allows to edit analysis configuration that is used from AnalysisManager, the configuration can have several profiles for analysis, that has to be handed over to the AnalysisManager

- n with mult answer opts is not number of interviewees

package at.fhv.qsys.analysis

analysis.xml -> no GUI support yet, XSD available

AnalysisManager: generate PDF, hand over (Zip)OutputStream, or it gets written to storage

- change on template analysis.xml affects all instances too (AnalysisConfigurationManager)

qsys-web

The two most important directories in the qsys-web project are the *src* directory and the *webroot* directory. The *src* directory holds all Java sources, while the *webroot* directory holds all static resources such as Javascript sources, CSS, XSL style sheets, XML files for application languages, images and documentation.

Java Resources

Server side resources are mainly split into the packages `org.qsys.quest.action.process` and `org.qsys.quest.action.view`. Therefore, user requests may demand a resource of the process or view subpackage. The process subpackage generally holds resources that require request parameters while the view subpackage generally holds resources that serve dynamic content independently of request parameters.

A resource of the process subpackage usually consists of a class inheriting from `BaseQsysProcessAction` and a class inheriting from `BaseQsysForm` while a resource of the view subpackage usually only consists of a class inheriting from `BaseQsysViewAction`. In both cases, the `BaseQsysForm` is the interface to request parameters.

Subclasses of `BaseQsysProcessAction` and `BaseQsysViewAction` have struts markup in the class body for page navigation. These markups are compiled into `WEB-INF/struts-config.xml` when building qsys-web. For updating page navigation the markup in the class body needs be changed. Changes in `struts-config.xml` are discouraged as they are overwritten at the next build.

XSL Transformation

For performance reasons XSL templates are kept in memory after first use. For the implementation see `XSLUtils.xsltTransform()`.

The mapping between above mentioned actions and XSL templates takes place in `qsys-web/webroot/WEB-INF/xslt-map.xml`. The XSL templates are stored in `WEB-INF/xsl/`. Language resources used by XSL templates are also configured in `xslt-map.xml` and the files are stored in `qsys-web/webroot/WEB-INF/lang`.

Javascript

Javascript sources are stored in qsys-web/webroot/js.

Tabbing

Project-Home: <http://www.barelyfitz.com/projects/tabber/>

Rich Text Editing

Project-Home: <http://tinymce.moxiecode.com/>

Hint for debugging: The header includes js/tiny_mce/tiny_mce.js, which is a copy of js/tiny_mce_src.js where all whitespaces are taken out. When trying to understand the program logic and when performance does not matter, it is more practical to include the readable source file.

Recommended steps for adding new feature

1. add feature to qsys-core and test
2. add feature to qsys-web and test

Bear in mind to keep responsibility in qsys-core.

Suggested Features

- Questionnaire import/export feature for migration between developer and production server.