

## Generic Java Reports Architecture

Vladimir Olenin

### Overview

The system creates reports in PDF and XLS formats and consists of two major parts: Database stored procedures (DB SP) and Java representation layer (JSP). The system is completely DB driven and currently allows up to 14 different input parameters from the user.

### User Cases

The system works in 2 steps. First from the web page user selects a report (s)he wants to create. After that the page is reloaded with the input parameters (any HTML input element) for the selected report. After setting report parameters user presses 'submit' button and PDF or XLS report would show up.

### Java Classes Overview

The Java layer is build using Struts framework.

#### JSPs

*reports/genericReports.jsp* – JSP with input parameters for the reports.

#### Classes

*com.aecl.proposal.web.GenericReports* -> Action – request processor for genericReports.jsp. Mapped to /reports.do path.

*com.aecl.proposal.web.ReportsBean* -> ActionForm – supporting data bean for genericReports.jsp

*com.aecl.proposal.service.ReportsService* -> BaseService – service to retrieve reports related data from the DB.

*com.aecl.proposal.entities.ReportPageModel* -> Object – generic data model for PDF/XLS reports.

#### PDF package.

*com.aecl.proposal.pdf.PDFReport* – generates PDF reports using iText library

(<http://www.lowagie.com/iText/index.html>)

*com.aecl.proposal.pdf.HeaderFooter* – supporting class for PDFReport to generate Header/Footer on the reports

#### XLS package.

*com.aecl.proposal.xls.XLSReport* – generates XLS reports using jakarta POI library

(<http://jakarta.apache.org/poi/hssf/index.html>)

## Architecture Description

The driving class for input parameters is ReportsBean. It contains all the data necessary to display input elements and store data input. Each parameter has 'visibility' flag which defines if the parameter should be visible for the chosen report or not. The rest of the data is for parameter data representation only. See ReportsBean class and genericReports.jsp for more details and as an example.

The visibility flags as well as initial data for input elements (eg, drop down lists) are retrieved (by reportNumber, which is either a default report or the user selection) from the DB in the ReportsService class. JSP would display only those parameters that have visibility flag set to 'true'. After the user set report parameters, the data for the report is retrieved from the DB. The parameters that were not visible on the page would be passed as NULLs. The DB returns report data together with some formatting flags which include:

- bold flag – for each row
- add blank line below – for each row
- column header – one per column
- column width – one per column
- column alignment – one per column
- format pattern string – one per column (applies to non-string datatypes, such as DATE, NUMBER)

Report data together with supporting information is passed to report generating classes which write pdf/xls data directly into output stream which makes report come up on client side.

## Adding New Report

Generally speaking, the only thing required to set up new report is creating all supporting structures in the DB. The Java representation layer would catch them up automatically.

NOTE: this is valid only if all 14 input parameters are already set up in the JSP and supporting servlet/service. Otherwise appropriate minor modifications (add new input in the JSP, add appropriate properties to ActionForm and pass these properties correctly to the SP) should be done to Java code.

NOTE: remember that right now the system is restricted to only 14 *unique* input parameters, which means that each of them always stands for the same property for all reports (basically, all reports share exactly the same parameters), but only required parameters would be displayed for each report and sent to the SP.

## Input Parameter Setup

To setup up an input parameter you should:

- 1) add appropriate properties to the ReportsBean (visibility flag + support data structures)
- 2) initialize these new properties for each 'load' request from the user from the DB in ReportService.initReportsBean method.
- 3) pass correctly the new input values to the DB in ReportService.getReportData method.

### Data Flow and Work Flow

