QuanTB. Developer’s guide

Оглавление

[About QuanTB 2](#_Toc488929329)

[Challenge 2](#_Toc488929330)

[Solution 2](#_Toc488929331)

[User Guide 3](#_Toc488929332)

[Audience 3](#_Toc488929333)

[Application 3](#_Toc488929334)

[Database 4](#_Toc488929335)

[Representation 4](#_Toc488929336)

[Metadata 4](#_Toc488929337)

[Quantification files 4](#_Toc488929338)

[Error logs 5](#_Toc488929339)

[How to start development 5](#_Toc488929340)

[Pre - conditions 5](#_Toc488929341)

[Development environment 5](#_Toc488929342)

[Project’s layout 5](#_Toc488929343)

[Command line parameters 6](#_Toc488929344)

[Development hints 7](#_Toc488929345)

[Top level packages 7](#_Toc488929346)

[Data Model (org.msh.quantb.model) 7](#_Toc488929347)

[Data structure 7](#_Toc488929348)

[Data Model 8](#_Toc488929349)

[Put it all together (org.msh.quantb.model.mvp.ModelFactory) 9](#_Toc488929350)

[Calculators and logic (org.msh.quantb.services) 9](#_Toc488929351)

[Calculators (org.msh.quantb.services.calc) 9](#_Toc488929352)

[The other classes in package org.msh.quantb.services.calc are responsible to particular and auxiliary calculation tasks or represent additional temporary data structures. 10](#_Toc488929353)

[Logic (org.msh.quantb.services.io) 10](#_Toc488929354)

[Excel import-export (org.msh.quantb.services.excel) 11](#_Toc488929355)

[Put it all together (org.msh.quantb.services.mvp.Presenter) 11](#_Toc488929356)

[Screen forms (org.msh.quantb.view) 11](#_Toc488929357)

[Swing and Beans Binding 11](#_Toc488929358)

[The main window (org.msh.quantb.view.window.MainWindow) 11](#_Toc488929359)

[Panels (org.msh.quantb.view.panel) 11](#_Toc488929360)

[Dialogues (org.msh.quantb.view.dialog) 12](#_Toc488929361)

[Other classes (org.msh.quantb.view, org.msh.quantb.view.tableExt) 13](#_Toc488929362)

[Put it all together (org.msh.quantb.view.mvp.ViewFactory) 13](#_Toc488929363)

[Test codes and data 14](#_Toc488929364)

[Test codes 14](#_Toc488929365)

[Test data layout 14](#_Toc488929366)

[How to make installer 15](#_Toc488929367)

[Why does installer need 15](#_Toc488929368)

[How do we create installers for Windows and Mac 15](#_Toc488929369)

[Tools 15](#_Toc488929370)

[Steps to create the installer 16](#_Toc488929371)

[MS Windows and Java 64 bits 16](#_Toc488929372)

# About QuanTB

## Challenge

Ensuring that patients have continuous access to tuberculosis (TB) treatment requires complex projections and calculations by TB program staff. These projections are becoming more challenging as new diagnostic tools rapidly increase the number of individuals diagnosed, and thus the quantities of medicines needed. As treatment regimens change, national programs must plan carefully for phasing in and out various medicines in order to manage the risk of stock outs. Frequent quantification, along with management of stock on hand, are vital to ensuring that appropriate types and quantities of medicines are available to meet the changing needs of programs and that TB programs are alerted of potential problems with TB medicines in a timely manner.

## Solution

To promote a systems-strengthening approach to TB medicines management, the USAID-funded Systems for Improved Access to Pharmaceuticals and Services (SIAPS) Program developed QuanTB—a downloadable, desktop tool that transforms complicated calculations into a user-friendly dashboard displaying key quantification and supply planning information. By alerting users to risks of stock outs, QuanTB operates as an early warning mechanism.

QuanTB allows you to forecast needs of tuberculosis medicines and treatment regimens for any period of time. Quantification can be done by:

• The number of enrolled and expected cases per month or by percentage of use of each medicine

• Either multidrug or single-drug regimens

For cases currently on treatment, QuanTB takes into account the month when treatment was started and the duration of regimen. The system also takes into account the medicines currently in stock (stock on hand), and the medicines on order (stock on order) during the quantification period. This tool also allows you to project your future procurement and budget needs.

You may download the QuanTB software from http://www.msh.org/resources/quantb

We would appreciate your testing of this tool under different scenarios and providing feedback, comments, and suggestions. Please send your comments to [quantb@msh.org](mailto:quantb@msh.org).

## User Guide

The User Guide explains installation process and usage of QuanTB. Please refer to this document first. In addition, it is highly recommended to install the latest version of QuanTB and then to calculate quantification from examples.

# Audience

This guide provides information that allows understand QuanTB from developer’s point of view. It is supposed that reader of this document is Java Software Developer.

Java Software Developer is very unclear term. It is because of huge amount of third party Java tools, that often calls “Java Ecosystem”. It is impossible to be good expert for all tools. Some specialization always takes place.

To develop QuanTB we have used restricted set of third party open source Java tools. Developer can self-assess skill for each tool by using the following gradations:

**Awareness** – means that developer can answer what is this tool and where it is used

**Knowledge** – means that developer can point to at least one training course of this tool, which the developer has taken/passed last 2 years.

**Ability** – means that developer can prove at least one successful project for which the developer has used this tool.

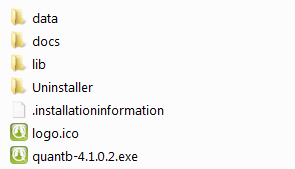
Desired skill requirements for each tool are below:

* Java SE development - ability
* Maven 2 – knowledge
* Beans Binding (JSR-295) – knowledge
* JFreeChart (<http://www.jfree.org/jfreechart/>) - knowledge
* Eclipse or Intellij IDEA or NetBeans - ability
* Java Swing - ability
* XML – knowledge
* Launch4J, IzPAck - awareness
* Version control systems, like GitHub, SVN - awareness

Requirements above are only for formal assessment. The ability and desire to learn new things is a key factor.

# Application

QuanTB is Java Desktop application that uses Java Swing for GUI and set of XML files as database. QuanTB requires Java JRE 64 bit version 1.6 and above[[1]](#footnote-1). For development purpose, is necessary to use appropriative JDK.

Typical directory layout of installed QuanTB on MS Windows is below. 

| **Component** | **Purpose** |
| --- | --- |
| Quantb-4.1.0.2.exe | Main application. It is “wrapped” by launch4J executable JAR file. Run time parameters will be described later. Contains all QuanTB specific codes. Part “-4.1.0.2” means version number. Can be different |
| logo.ico | Splash screen |
| .installationinformation | File created by the installer. Contains necessary data for uninstaller. |
| Uninstaller | Folder, contains uninstaller (uninstaller.jar) |
| lib | Third – party open source Java libraries. |
| docs | Default folder for Quantification data files (qtb). Just after installation contains only examples. As a matter of fact, user can use any folder to store Quantification data. |
| data | Folder that contains dictionaries, user manual and templates. In short – system metadata. |

# Database

## Representation

The database consist of Metadata, Quantification data and Error logs. The database represents as a set of XML files unrelated each other.

## Metadata

Metadata uses by QuanTB solely to create new Quantification files and/or new metadata. Quantification does not depend on metadata. It means that a change in Metadata does not affect any given Quantifications. For instance, you can change a medicine definition in medicines metadata. Nevertheless, it will not affect regimens that use this medicine and Quantification that uses this medicine.

Metadata files are following:

|  |  |
| --- | --- |
| **File** | **Purpose** |
| med\_dictionary.xml | Medicines that is used to determine treatment regimens. |
| reg\_dictionary.xml | Treatment regimens is used for new Quantifications |
| locale.xml | Current language and saved path to folder from which last Quantification has been loaded. |
| history.xml | Last five Quantifications that have been loaded |

## Quantification files

Quantification files are XML format files with extension “qtb” like “myquantification.qtb”. Any given quantification file is a self-sufficient Quantification. It means that users can send quantification files each other regardless on metadata.

## Error logs

Under folder, “data” you may find folder “log” This folder contains XML format files with names like 2017-05-22err.qtb. Any given file contains information about one run-time error. This information intends solely for QuanTB tech support.

# How to start development

## Pre - conditions

QuanTB is a “pure” Maven 2 project. It means that any convenient development tool such as Eclipse, Netbeans, IntelliJ IDEA, even Notepad may be used. Oracle Java JDK version 1.6 and later must be installed. We are using the latest JDK without any problems. Seems it is possible to use Open Java, however we did not test it.

You do not need any additional development tool such are RDBMS and/or application server. Ensure only that you have enough hardware, at least:

1. Processor – Intel I3, any generation
2. RAM – 8 GB
3. Disk (HDD, SSD) – 10 GB free

## Development environment

You may download source code of the QuanTB from code repository such are GitHub or SVN or receive them as project archive. For simplify, assume that you have project archive and you going to develop it in Eclipse. We have developed this project in Eclipse Mars. Seems as not restricted to develop this project using the latest Eclipse version.

Because QuanTB is Maven 2 project, development environment installation is simple. Unpack project archive somewhere to local disk and import as Maven project. Please, do not forget run Maven-Install on it.

## Project’s layout

| **Name of file, folder** | **Description** |
| --- | --- |
| quantification | QuanTB Maven 2 project directory |
| src | Standard Maven 2 source folder |
| src/main | Main codes and resources |
| src/main/java | Java source codes |
| src/main/resources | Resources such are:   1. XSD schemas 2. Images 3. I18 string resources |
| src/test | Test codes and resources |
| src/test/java | JUnit test source codes |
| src/test/resources | Metadata for JUnit tests and development environment as well as some JUnit test specific files. Acts like “data” folder for development |
| src/test/resources/doc | Acts like “doc” folder for development. Storage for Quantifications, exported Excel etc |
| src/test/resources/doc/byJUnit | Acts like “doc” folder, solely for JUnit tests. Please, do not change something in this folder, without corresponding changes to JUnit tests |
| src/test/resources/log | Error logs. Please, clean this folder periodically. |
| target | Standard Maven 2 target folder. Contains QuanTB applications created by Maven Install task. There are two representation of an application:  quantification-*version*.jar  quantb-*version*.exe  Both representations are identical from functionality point of view. Representation as exe file is jar representation wrapped by Launch4J |
| target/classes | Translated Java codes. You do not need them |
| target/lib | Third party libraries. You must distribute them along with the application |
| target/maven-archiver | Something created by Maven |
| target/surefire-reports | JUnit tests reports |
| target/test-classes | Translated Java codes. You do not need them |
| pom.xml | Maven 2 control file |

## Command line parameters

Main class is org.msh.quantb.services.mvp.Presenter. Specific command line parameters are:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Purpose** | **Default value** |
| -Dquantb.data.path | Metadata folder | src/test/resources |
| -Dquantb.doc.path | Default folder to store Quantification files | src/test/resources/doc |
| -Dquantb.version | Current version to display as well as check compatibility of Quantification files | Development mode |

As you can see, default values suit only for development. Therefore, for production all command line parameters are mandatory. In contrary, for development you should omit them.

For example:

For MS Windows, launch4J (http://launch4j.sourceforge.net/) configuration contains in pom.xml such definitions of these parameters:

<opts>

<opt>

-Dquantb.data.path="%EXEDIR%\\data"

</opt>

<opt>

-Dquantb.doc.path="%EXEDIR%\\docs"

</opt>

<opt>

-Dquantb.version=${project.version}-${timestamp}

</opt>

</opts>

Command line for Mac OS:

java -Dquantb.data.path="data/" -Dquantb.doc.path="docs/" -Dquantb.version=4.1.0.2-20170504 -Xms512M -Xmx2G -jar quantification-4.1.0.2.jar

# Development hints

## Top level packages

Source codes are divided by three main packages:

|  |  |
| --- | --- |
| **Package** | **Purpose** |
| org.msh.quantb.model | Java API for XML Binding (JAXB), version 2+, generates to this package Java codes that are Data Model. Sources are XML schemas from “src/main/resources” |
| org.msh.quantb.services | Calculators, validators, controllers and Excel related codes |
| org.msh.quantb.view | Swing UI related codes |

## Data Model (org.msh.quantb.model)

### Data structure

Current POM file contains such JAXB 2+ Maven plugin configuration:

<plugin>

<groupId>org.jvnet.jaxb2.maven2</groupId>

<artifactId>maven-jaxb2-plugin</artifactId>

<version>0.8.2</version>

<executions>

<execution>

<id>1</id>

<phase>generate-sources</phase>

<goals>

<goal>generate</goal>

</goals>

</execution>

<execution>

<id>2</id>

<phase>test</phase>

<goals>

<goal>generate</goal>

</goals>

</execution>

</executions>

<configuration>

<schemaDirectory>src/main/resources</schemaDirectory>

<generateDirectory>src/main/java</generateDirectory>

<includeSchemas>

<includeSchema>DataTypes.xsd</includeSchema>

<includeSchema>MedicineDic.xsd</includeSchema>

<includeSchema>RegimenDic.xsd</includeSchema>

<includeSchema>Forecast.xsd</includeSchema>

<includeSchema>ErrorLog.xsd</includeSchema>

<includeSchema>Locale.xsd</includeSchema>

</includeSchemas>

</configuration>

</plugin>

Folder “src/main/resources” contains such schemas

|  |  |
| --- | --- |
| DataTypes.xsd | This schema contains definitions of common complex XML types that are using in other schemas |
| MedicineDic.xsd | Medicines Dictionary |
| RegimenDic.xsd | Regimens Dictionary |
| Forecast.xsd | Contain definitions of Quantification related complex types and Quantification itself |
| ErrorLog.xsd | Complex types for errors logging and Error Log itself |
| Locale.xsd | This schema represents data that stores between QuanTB runs. Such are Language, the most recent folder with Quantification data, etc. Only for user’s convenience. |

### Data Model

Data Model is Java source codes that intend to access metadata and Quantifications. These source codes are under package “org.msh.quantb.model”. Sub – packages are:

* errorlog
* forecast
* gen
* locale
* medicine

Maven JAXB 2+ plugin automatically generates Data Model in accordance with following rules:

1. Any given schema is bound to one and only one source code package.
2. Any given package contains file package-info.java.
3. File package-info.java contains annotation “@javax.xml.bind.annotation.XmlSchema”.This annotation determines which schema is bound to this package.
4. Any given source code package consist of following auto-generated classes:
   1. Java classes for XML schema complex types;
   2. Class ObjectFactory, contains codes to create instances of Java classes for XML types

### Put it all together (org.msh.quantb.model.mvp.ModelFactory)

Package org.msh.quantb.model.mvp contains class ModelFactory. It is not auto – generated class. This class intends to serve all data operations on whole Data Model. From Data Model point of view, this class implements Create, Read and Update operations[[2]](#footnote-2).

## Calculators and logic (org.msh.quantb.services)

### Calculators (org.msh.quantb.services.calc)

This package contains codes that intends for calculation both results of Quantification:

* Forecasting reports
* Orders

It is not surprise that this package contains many classes. Quantification is complex process. It will be a good idea to start your journey from class ForecastingCalculation. Entry point to it is method “execute”.

As you can see in this method, we have to:

1. Validate forecasting data
2. Clear all previous results
3. For each day of Quantification period calculate:
   1. Prognosis for cases on treatment quantity for any given day in Quantification period, based on
      1. Enrolled cases for each regimen (calcCasesOnTreatment)
      2. Expected cases for each regimen (calcNewCases)
   2. Needs for a medicine for any given regimen, based on regimen’s definition and cases calculated on previous step (calcMedicinesRegimes)
   3. Adjust results in accordance with data in “Adjust medicines quantity for attrition” (adjustResults)
   4. Needs for any given unique medicine from all regimens (calcMedicines)

At last, QuanTB imitates daily consume process to determine how much additional medicines we will need for any given day of Quantification period.

All above is enough for Forecasting reports. Forecasting result may be roughly represented as a list of such records (one record per day/medicine):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Medicine** | **Consumption (units)** | **Existed Quantity (units)** | **Needs for replenish**  **(units)** |
| Jan 19, 2017 | Amx/Clv(250/125) | 10 | 1 | 9 |
| Jan 20,2017 | Amx/Clv(250/125) | 20 | 0 | 20 |

For detail representation of forecasting result record, please refer to org.msh.quantb.model.forecast.ForecastingResult.

QuanTB uses forecasting result to build Forecasting reports. However to calculate orders we will need additional constraints:

1. Schedule of orders:
   1. All at once
   2. Annually
   3. Bi-Annually
   4. Quarterly
   5. Monthly
2. Minimal and Maximal stock constrains.
3. Pack sizes, because orders should be calculated in whole packs.
4. Adjustment coefficients to increase or decrease ordered quantity in accordance with expert judgement.

In addition, we have to take in account medicines prices and additional costs values.

Class “OrderCalculator” is responsible to calculate orders. Source data to calculate orders are Forecasting result, constrains and prices. Entry point to “OrderCalculator” class is method “execute”

As you can see in this method, we have to:

1. Calculate necessary deliveries based on constrains 1-4 (LogisticCalculatorPessimist[[3]](#footnote-3).exec)
2. Calculate additional order’s parameters such are costs (calcMedTotals)
3. Finally, calculate accelerated and regular orders (calcOrders)

In most cases, when constrains will became changed by a user, we will not need to recalculate Forecasting. For such cases, class OrderCalculator contains method “reExecute”.

### The other classes in package org.msh.quantb.services.calc are responsible to particular and auxiliary calculation tasks or represent additional temporary data structures.

### Logic (org.msh.quantb.services.io)

QuanTB implements complex logic to control data input and data changes. Classes from package “org.msh.quantb.services.io” implement this logic.

To control data changes QuanTB uses old good “java.beans” technology (<https://docs.oracle.com/javase/tutorial/javabeans/>). The most classes in this package are inherited from class AbstractUIAdapter that provides “java.beans.PropertyChangeListener” and “java.beans.PropertyChangeSupport” implementation. These classes mostly are decorators (https://www.tutorialspoint.com/design\_pattern/decorator\_pattern.htm) for corresponding Data Model classes. For instance, class ForecastUIAdapter is a decorator for Data Model class Forecast.

A decorator adds change control ability to any given Data Model class. This ability uses for data binding with on screen forms, data validation and calculation logic. In addition, a typical decorator contains equals and compare logic.

The data binding feature allows easy synchronize data from user input to on screen forms with corresponding Data Model. Eclipse development environment provides Swing Data Binding tools. You can find description of it at <http://help.eclipse.org/mars/index.jsp>, Chapter “WindowBuilder Pro User Guide”, Sub Chapter “Data Binding - Swing”. For Eclipse users, Eclipse – specific library beansbinding-1.2.1.jar is included as dependency to pom.xml.

Quantification’s initial conditions are complex. Therefore, data validation rules are complex too. The program needs to verify input data mostly “on fly”. Some validation codes you can find in special class ForecastUIVerify. The rest of validation codes you can find in decorator classes or in on screen form classes.

Changes of some Quantification’s initial conditions have impact on calculation or data input logic. It means that when some parameter changes, a callback function that implements change logic must be called. For instance, when a user make changes to end date of Quantification, result of previous calculation should become inaccessible for a user. Most of the callbacks you can find in “org.msh.quantb.services.mvp.Presenter”.

### Excel import-export (org.msh.quantb.services.excel)

Excel import and export functions use Apache POI library. Currently format XLSX is using. Only for example, we leave in source codes old outdated XLS processing by JXL package. Please, do not use them.

### Put it all together (org.msh.quantb.services.mvp.Presenter)

Main class of QuanTB is Presenter. This class contains method “main” to start QuanTB application and many callbacks to process system events like “user press a button” or “user type some data”. Class “Presenter” intends to orchestrate data model, on screen forms, services and calculators. Please, put your attention first on such methods of this class:

* main – start QuanTB
* Callbacks:
  + createForecasting – create new empty quantification data
  + openForecastingDocument – open existing Quantification data
  + runForecastingCalculation – run Quantification and display results
  + saveActiveForecasting – save Quantification data

Really, Presenter class looks like “bloated” controller in MVC architecture terms. In addition, archaic static methods do not add extra beauty to it. Seems, as it is time to deep refactoring…

## Screen forms (org.msh.quantb.view)

### Swing and Beans Binding

QuanTB uses standard Java Swing (<https://docs.oracle.com/javase/tutorial/uiswing/>) components with Substance 7.3 look and fill (<http://insubstantial.github.io/insubstantial/substance/>).

For data binding management, we uses Java Beans Binding Eclipse tool beansbinding-1.2.1.jar that mentioned above. Eclipse “WindowBuilder Pro” provides additional convenient tool called “Binding”.

Unfortunately, this tool is old and developers abandoned the project (<https://www.jcp.org/en/jsr/detail?id=295>). It means that nobody can guarantee that this tool will work in future releases of Eclipse. Fortunately, “JSR 295: Beans Binding” specification is alive.

Mainly, Swing components is bond to classes from “org.msh.quantb.services.io” package. In cases when it hardly or impossible, Swing components is bound to proprietary Swing models classes. For charting only specific model classes are using.

### The main window (org.msh.quantb.view.window.MainWindow)

This class contains Main Window definitions. First, put your attention on “menuBar” that implements main system menu. Second, put your attention on “panelDocuments” – tabbed panel that implements on screen layout of QuanTB.

### Panels (org.msh.quantb.view.panel)

Panels represent Quantification initial conditions and results, such are Forecasting reports and Orders. To easy find proper class for on-screen panel, please use following guide.

Main set of tabs in QuanTB looks like



Table below depicts relation between tabs and classes in packet org.msh.quantb.view.panel. Method showCalculationDetailsTabs of ForecastingDocumentPanel is a good start to explore.

|  |  |  |
| --- | --- | --- |
| **Primary tab** | **Secondary tab** | **UI Class name** |
| Parameters |  | ForecastingDocumentPanel |
| Summary |  | SummaryPanel |
| Medicines Report |  | MedicineConsumptionPanel |
| Cases Report | Treatment Regimen | CasesOnTreatmentPanel |
| Medicines |
| Medicines Detailed Report |  | DetailsPanel |
| Order and Schedule | Quantity and Costs | ForecastingTotalPanel |
|  | Additional and Total Costs |
|  | Schedule |
| Graphs | Stock Status | DashBoardPanel |
|  | Dashboard | StockGraphPanel |

As you can see, ForecastingTotalPanel is complex, therefore require additional explanation. Method “setAndBind” of ForecastingTotalPanel is a good start to explore.

|  |  |
| --- | --- |
| **Tab** | **How it works** |
| Quantity and Costs | Three JTables, Beans Binding is using |
| Additional and Total Costs | Three objects of FinalCostPanel class |
| Schedule | Flow of objects of DeliveryOrderPanel class |

Classes DashBoardPanel and StockGraph panel are complex too. To build graphs, these classes use JFreeChart 1.0.13 (<http://www.jfree.org/jfreechart/>). Please, start explore these classes from methods “setData”.

### Dialogues (org.msh.quantb.view.dialog)

Dialogues serve many aspects of data input. QuanTB program initiate dialogues by appropriative buttons and/or hotspots. Table below is reference for quickly find any given dialogue.

| **Dialogue** | **Purpose** | **Calls from** |
| --- | --- | --- |
| AboutDlg | About QuanTB box | MainWindow -> Main menu-> About |
| ForecastFileHistory | Allows select 5 recent files | MainWindow -> Main menu -> <Last 5 docs> |
| ForecastingBatchDlg | Add or edit medicine batch data on Stock Of Medicine parameters sub tab | ForecastingDocumentPanel, Stock Of Medicine sub tab |
| ForecastingOrderDlg | Add or edit medicine order data on Stock Of Medicine parameters sub tab | ForecastingDocumentPanel  Stock Of Medicine sub tab |
| ForecastingWizardDlg | Initiate a new Quantification | MainWindow -> Main menu-> File->New |
| ForecastSliceDlg | It allows input data to divide Quantification | MainWindow->Main menu->Divide&Merge-Divide quantification |
| MedicineDlg | Add new medicine to Medicines Dictionary or Edit existed one. | Buttons New and Edit on MedicinesDlg |
| MedicinesAdjustDlg | Allows adjust medicines quantity before Quantification calculation | ForecastingDocumentPanel, Parameters tab, button “Adjust medicines quantity for attrition” |
| MedicinesDecodeDlg | This dialogue uses for stock of medicines import. Allows to set relation between medicines going to import and medicines existed in Medicines Dictionary | MainWindow -> Main menu -> Import & Export -> Import From Excel ->Import Stock Data |
| MedicinesDlg | It represents list of medicines from Medicines Dictionary | MainWindow->Main menu->Medicines & Regimens->Edit medicines |
| MedicineSelectDlg | It allows to select medicine for single drug regimen | Button “Add Medicine” in PhaseComponent class. This component is part of RegimensDlg |
| MedicineSelectMulDlg | It allows to select medicine for multi drug regimen | Button “Add Medicine” in PhaseComponent class. This component is part of RegimensDlg |
| RegimensDlg | It allows add new regimen to Regimens Dictionary or edit existed one. In addition, same dialogue is using to display regimen details. | Buttons “New” and “Edit” on RegimensListDlg.  ForecastingDocumentPanel.  Hot spots in column Treatment Regimens on sub tabs “Enrolled cases” and “expected cases” |
| RegimenSelectDlg | It allows include or exclude regimens to/from Quantification | ForecastingDocumentPanel, button “Select treatment regimens” |
| RegimensListDlg | It represents list of regimens from Regimens Dictionary | MainWindow->Main menu->Medicines & Regimens->Edit treatment regimens |
| SelectMergeDlg | It allows select Quantifications to merge | MainWindow->Main menu->Merge quantification |
| WrongVersionDlg | This dialogue displays when version of Quantification file going to open is incompatible with current version of QuanTB | After MainWindow->Main menu->File-Open… |

### Other classes (org.msh.quantb.view, org.msh.quantb.view.tableExt)

The remainder of classes in packages “org.msh.quantb.view”, “org.msh.quantb.view.tableExt” are renderers, converters, helpers and other Swing related staff.

### Put it all together (org.msh.quantb.view.mvp.ViewFactory)

Class ViewFactory serves creation of panels and dialogues. Object with class “ForecastingDocumentPanel” is stored here.

In addition, this class contains customizations of standard Swing dialogues as well as codes for load and reload Main Window.

# Test codes and data

## Test codes

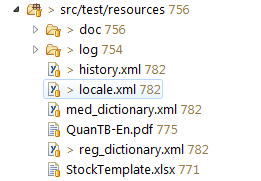
Test codes are in “src/test” folder. QuanTB currently uses only JUnit 3 tests. The mandatory units for tests are calculators. Please, put your attention on class “org.msh.quantb.services.calc. TestCalculation”. Test data resides in “src/test/resources/byJUnit”. Please, make only conscious changes in this folder and never use data from it for other types of tests; such are integration and system testing.

Other test codes such are Excel and Utilities are optional.

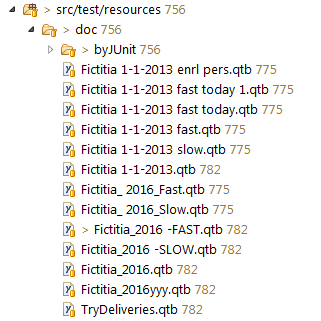
## Test data layout

All test data resides in “src/test/resources” folder. This data uses for both development/debugging and unit tests.

The root contains all dictionaries with test data to use for start QuanTB in debug/development mode. This folder is like folder “data” for real installation.



Folder “src/test/resources/doc” contains test QuanTB data for debug/development mode. This folder is like folder “doc” for real installation. Typically, this folder contains Quantification examples



These files are for integration and system testing, therefore you can change it freely. Sub – folder “byJUnit” is for JUnit tests and you cannot change any file in this folder without JUnit tests change.

Folder “log” contains error logs. These errors occur in development/debug process, therefore:

* There are many files here
* These files are useless

Please, do not forget periodically remove them.

# How to make installer

## Why does installer need

QuanTB is “pure” Java application. Therefore, QuanTB has not be installed on Operating System level. Nevertheless, it does not mean that installer is not needed.

Installer is needed because:

* Right Java must be installed on user’s computer
* QuanTB has complex folder layout
* User should has read/write access to QuanTB folder

In addition, Installer is familiar way to add new software for plain users.

## How do we create installers for Windows and Mac

### Tools

To build installers we use IzPack tool version 4.3.5. Why so outdated? Because it works. Fill free to use the latest version (<http://izpack.org/>). IzPack is cross – platform tools and well suit for both Windows and Mac.

Following consideration to create installer configuration are important:

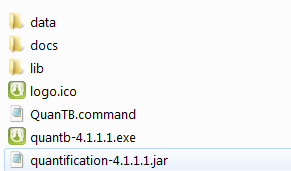
1. Minimal Java version should be 1.6, maximal 1.8
2. Target folder to install should be sub folder of "$USER\_HOME”, for instance "$USER\_HOME\QuanTB"
3. Installation packs should follow such rules:
   1. Files “med dictionary.xml” and “reg\_dictionary.xml” should be copied from Installation to “$INSTALL\_PATH/data” folder only when a user allows this operation. These files are necessary for QuanTB for first installation; however following installations (upgrades) must not rewrite user’s changes in these files
   2. Examples of Quantification should be copied from installation to “$INSTALL\_PATH/doc” folder only when a user allows this operation. Considerations are same as above.
4. User Guide should override the old one in “$INSTALL\_PATH/data” folder unconditionally.
5. Excel template for import data (StockTemplate.xlsx) should override the old one unconditionally in “$INSTALL\_PATH/data” folder.
6. All contents of “$INSTALL\_PATH/lib” folder should be overridden unconditionally
7. File “logo.ico” should be overridden unconditionally for both “$INSTALL\_PATH” and “$INSTALL\_PATH/data” folders
8. For Windows installation file “$INSTALL\_PATH /quantb\*.exe should be overridden unconditionally
9. In addition for Mac installation
   1. Files “quantification\*.jar”, “QuanTB.command” should be overridden unconditionally to “$INSTALL\_PATH”:
   2. For “$INSTALL\_PATH/ QuanTB.command” should be sat executable privilege “754”
   3. For folders “${INSTALL\_PATH}/data” and “$INSTALL\_PATH/docs” should be sat executable privilege “666”

### Steps to create the installer

Command “maven:install” creates two executable files – “quantification-xxxxxx.jar” and “quantb-xxxxxx.exe”. In addition, this command copies right libraries to project’s folder “target/lib”

File quantb-xxxxxx.exe is quantification-xxxxxx.jar wrapped to MS Windows executable format by launch4j tool (<http://launch4j.sourceforge.net/>).

It will be a good idea to create somewhere on disk a release from which installer will be created. On my disk, release looks like below.



This release contains both Windows and Mac executables as well as script QuanTB.command to run QuanTB on Mac computer. Content of this file is below

cd "$(dirname "$0")"

java -Dquantb.data.path="data/" -Dquantb.doc.path="docs/" -Dquantb.version=4.1.1.1-20170623 -Xms512M -Xmx2G -jar quantification-4.1.1.1.jar

killall Terminal

This release is good start point to create installers. Steps are below:

1. Run IzPack twice with different configurations to create installers for Windows and Mac. Results are two executable jar files. These files are installers. It is possible to prepare one universal installer in jar format; however, this job is for future.
2. Run launch4j to create Windows executable installer from jar installer. It is because:
   1. For the most plain Windows users, Windows executable as installer is familiar
   2. Windows executable that created by launch4j can test Java installation on user’s computer.

## MS Windows and Java 64 bits

QuanTB program requires Java 64 bits runtime. It is not a problem for Mac users. However, it is not trivial task for plain user to install Java 64 bit on Windows computer. Regular way to install Java from java.com will install 32 bit version in case when user’s web browser is 32 bit.

There are such solution possible

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Solution** | **Advantages** | **Disadvantages** |
| 1 | Provide a link to a right version on the Oracle site | The easiest way | Oracle may change download address without notification |
| 2 | Provide a link to a proven version on our site | As easy as #1 | Nobody can change download address, however it is hardly to provide same download performance as Oracle does |
| 3 | Provide a warning message like "Java must be installed or updated on your computer before you can install QuanTB. Please go to www.java.com to download and install the latest version. You must use Chrome or Edge (not Firefox) to do this. When you are finished installing Java, start the QuanTB installation again." | The fastest and legal way to install Java | Bit more complex. In addition, once upon a time the latest version of Java will be 1.9 that might will be incompatible with QuanTB codes |
| 4 | Include Java Runtime to QuanTB installation | Seamless for user. Any already installed Java will be suit for installer | QuanTB installation and runtime codes will grow at least in four times. |

Seems, as Solution 3 is preferable.

1. Currently we have not full specifications for Java 1.9 [↑](#footnote-ref-1)
2. Delete operation is excess. QuanTB is using Update instead. [↑](#footnote-ref-2)
3. Hereafter, it is possible to apply different strategies to calculate deliveries. Currently we are using only “worst case” strategy. [↑](#footnote-ref-3)