**ChalkServer Documentation**

**SVN Repo path: svn://svn.bns/equitytrading/trunk/PortfolioDesk/ChalkTalk**

**Production path:** [**\\t65-w7-eqcash\incoming\ChalkServer\htmlBuilder\ChalktalkServer-1.0.jar**](file:///\\t65-w7-eqcash\incoming\ChalkServer\htmlBuilder\ChalktalkServer-1.0.jar)

This is written in java, set it up in Eclipse.

This server handles report builds requests from Flow and Order Tracker. It builds the html (and excel file for the block report) and calls an email program that emails the report out. It runs on the eqcash machine and restarts everyday at 1130PM. To run the server on your own machine, run the main function in Application.java (server pkg). All the web request mappings are in ChalktalkController.java. When a request is received, one of the service functions in the singleton engine is called.

Post Trade report (/send-PostTradeReport)

* Generates post trade pdf, request sent by Flow (if you need details ask Bill)

Block report builder (send-BlockReport, send-morning-BlockReport)

* Both builds the block report. Requests can be sent through Order Tracker or Flow. The send-morning-BlockReport is triggered by an automated process at 6AM and 12PM. The 6AM process that takes the block information from yesterday (in Order Tracker) whereas the process at 12PM takes information from Flow. The 6AM process creates a general one that includes all industries, and for each of the liability traders who covers different sectors (Rusty, Scott, Paul, Colin). The automated process is on the eqcash machine as standalone .js files (run using node). They are in: C:\MorningBlockList.
* Builder creates the HTML format of the email, as well as an excel file to be attached to the email

News report builder (send-Chalktalk, send-Energy-Insights)

* Send-Energy-Insights is to handle specifically the energy insights report, while send-Chalktalk handles all other types of reports (Chalktalk, Morning, Halftime, etc)
* The web request passes in pertinent parameters for building the report, including the name of the person making the request, the type of report (indicated by id), the news items selected, and stock-specific information such as return and excess volume, etc.
* For the chalktalk report, the sections that want to be included are also passed in as a parameter
* In the engine, the list of comments is sorted by their ranking from the Flow page and then the appropriate report builder is called according to the type of report requested
* Several reports require ticker-specific data from scotiaview.com, which is the Scotiabank equity research site. Specifically the target price, rating, and link to the most recent research report are scraped from this website in the ScotiaViewParser.java (html pkg).
  + Every time the site is accessed the connection might time out so it tries for up to 5 times
  + First the cookie for login into the site is saved, then it obtains the securities map (which maps ticker and exchange to a securityID). The URL for Scotiaview for each ticker is formed using the securityID
  + The report builders initialize an instance of the parser, then for each stock on which there is a news item, it obtains the scotiaview information by calling the instance method getSymbolResearch (ric). Sometimes there is no research on for the ticker on the exchange specified by the ric, in which case we look to see if it is interlisted and check if the interlisted stock has research on scotiaview. If there is no research it just returns “N/A”

Overview of Chalktalk Report Builder (ChalktalkReportBuilder.java, under html package)

* The other news report builders are very similar (or simpler) in design so will go through this one in detail
* Chad and Tim use this for their before open morning report (~845AM) after all the comments have been loaded on Flow
* The idea is that there is a main HTML (ChalktalkTemplate.html) which serves as the ‘frame’ of the report. The different sections/parts are placed in programmatically, and is denoted by curly brackets. For example {{bellImage}} is replaced by the actual location of the image.
* For this particular report there are three images that change daily: Bell.png, Marb.png, Revisions.png, which Chad drop in the location: [\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages](file:///\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages)
* See below section “Note on the Images in the reports”
* For this report the three images are hosted on GitHub. So they are moved to another folder, timestamped, and committed to the remote repo
* It then gets the general before the bell, index events comments by making a call to the Flow Server.
* Method calls that create different parts of the report return the corresponding HTML, for example “buildBellComments” takes a HTML template for the news in “Before the Bell” section and fills in the needed info
* Method *includeSelectedSections* ensures only requested sections are in the report
* There are some weird characters stored in the database that may not be compatible so they are replaced using a method in the Utilities static class. If you see any weird characters in the report just find out what they are and add it to this method.
* After the template HTML is replaced with the actual content, it calls a “MailRequest.sendRequest()”, which calls the emailing program on Nitin’s server (scvmapp508) located D:/Tasks/MailService/SendMail.jar. Only Nitin’s server has the authorization to send emails. The email program takes a config file as the parameter. The config file specifies the subject, email addresses of sender and recipients, the path of the body of email as an html format, and any attachments
* Each person has their own signature. You have to create a custom HTML for each person’s signature. The location of the file is mapped by the “users.json” file. The users.json file also specifies to whom the report is sent to.

Note on the Images in the reports

* For most reports, we use github links for the images (the Scotiabank logo, different charts, etc)
* The login credentials for the github:
  + Login: chalktalkimages
  + Password: abcd123
* The repo is named Chalkimages
* For Chad’s report (the Chalktalk), everytime request to create/email Chalktalk report is received, a git commit takes the images in this folder: [\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages](file:///\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages) And copies them over to the ‘archive’ subfolder with a timestamp on each one … these timestamped images are then embedded in the report
* There is also a way to ‘update’ these timestamped images (for all of today’s images only) that are already embedded in the emails (because files hosted on github have URLs that do not change once they are added) by running ‘UpdateTodaysImages.bat”, which runs the java executable UpdateTodaysImages.jar
* For the images in the other reports, the images are taken directly from the directory [\\t65-w7-eqcash\incoming\ChalkServer\DailyScoop](file:///\\t65-w7-eqcash\incoming\ChalkServer\DailyScoop), or \GeorgeChalktalkImages
* To get rid of images that were more than 30 years old in the archive folder AND sync this change to the remote repo on Github, run RUNclearOldChalkImagesArchive.bat in [\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages](file:///\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages)
* To commit changes and sync to remote repo on Github in general, run SyncImages.bat in [\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages](file:///\\t65-w7-eqcash\incoming\ChalkServer\Chalkimages)
* Note that some of the images in Chalkimages are no longer in use by any report.

Server Status (/get-Server-Status)

* There is a hashmap that keeps track of on which step the report builder is on for each user, to keep the user informed, this is because the ScotiaView parsing step takes a lot of time.

Compiling source code into a jar (java executable):

* This is a Maven project, you need to have Maven installed. The jar is compiled via command line: ‘mvn install’
* The compiled jar will be in the subfolder: target
* Open the jar using WinRAR/ZIP, in the ‘lib’ folder, delete log4j-over-slf4j-1.7.12.jar
  + This is an issue because there’re two loggers, we need to remove one of them or else there will be a runtime error

HTML templates, signatures, and user.json file

* The most updated/complete versions are in the prod path
  + [\\t65-w7-eqcash\incoming\ChalkServer\htmlBuild](file:///\\t65-w7-eqcash\incoming\ChalkServer\htmlBuild)
* Every time source code is changed and new jar is compiled, place the new jar into the above path
* If there are some issues, check the log in the subfolder “log” …
* To debug/test, run the chalkserver locally, and run the FlowClient locally. Make REST calls to the local chalkserver. Comment out the emailing part and inspect the output HTML.