APPENDIX A

SOURCE CODE

This appendix contains all the source code I produced in the project. Below is a list of the main files and what their function is in the product.

APPENDIX B

INSTALLATIONS

This appendix contains the installation procedure. There are two different installation procedures. One which handles how I install on the Freyr server, and another how a development server can be installed for demonstration purposes.

Installation on server

This document describes the installation on Freyr's development server.

Python libraries

Python libraries are required in order to run the application. The easiest way to install is to use a tool called EasyInstaller, which downloads libraries from the internet. The following describes the procedure for the required libraries.

1. First, EasyInstaller must be installed, it is in the ”installer” folder.
2. After it is installed, then easy_install can be used.
3. easy_install lxml==2.1.1 BeautifulSoup
4.

Working on server

This is how I should work with the Freyr development server. I am using my own hostname yani.dk to point to the development server.

Connect to the server with Putty:
Hostname: yani.dk
Username: kereena
Password: .....

Installing on server

The following steps installs the application once Django and required libraries are installed:

1. Get the source code from subversion. This is ONLY first time.
   svn co http://svn.kokila.it/repo/sandbox/StockScreener/stockscreener stockscreener
2. Update the source code from subversion:
   cd stockscreener
   svn update
3. Create database:
   python manage.py syncdb
4. Load database dump:
   python manage.py loaddata graph_test.json
5. Load symbols:
   python manage.py updatesymbols

6. Start the server.
   See the “Starting application on Freyr server” step.
7. Load url in browser:
   http://yani.dk:7400/admin/ - Admin interface

Starting application on my PC

The program has to run little different depending on where I start it. This is because there is a Microsoft ISA server at the VejleHS network, which requires NTLM (Microsoft proprietary protocol) authentication, which Python does not understand.

Starting at vejlehs

1) Start the ntlmaps proxy server. Go into its folder and double-click on the runserver.bat
2) Start the stockscreener application. Click on the “manage.py runserver” shortcut on the desktop.

Starting from home

1) Start the stockscreener application. Click on the “manage.py runserver” shortcut on the desktop.

Starting application on Freyr server

For working on the server, first I have to follow the installation steps. On the server there are two scripts that I can use for starting and stopping the server.

1) Connect to “yani.dk server” using Putty
2) Try to open the old server process: ./continue.sh
   1. If it is success, I should see the output from the server. I can then press Ctrl-C to stop the server. After this I see the message:

      “Screen terminated”. - I press Ctrl-C and continue to step #3
   2. If it is failure (server not running) I see the message:

      “There is no screen to be resumed matching stockscreener” - I can continue with step #3
3) Start the server process: ./start.sh
   1. If it is success, I should see a message that server is running and that I can press Ctrl-C to stop the server.
   2. I can now shutdown the Putty application.
   3. If it is error “screen terminated”, it means that I did not try step #2.
   4. If it is another error, and server won't start, I have a problem with my code, and I need to look at how to fix it.
4) Done, check that I can view the URL: http://yani.dk:7400/search/

APPENDIX C

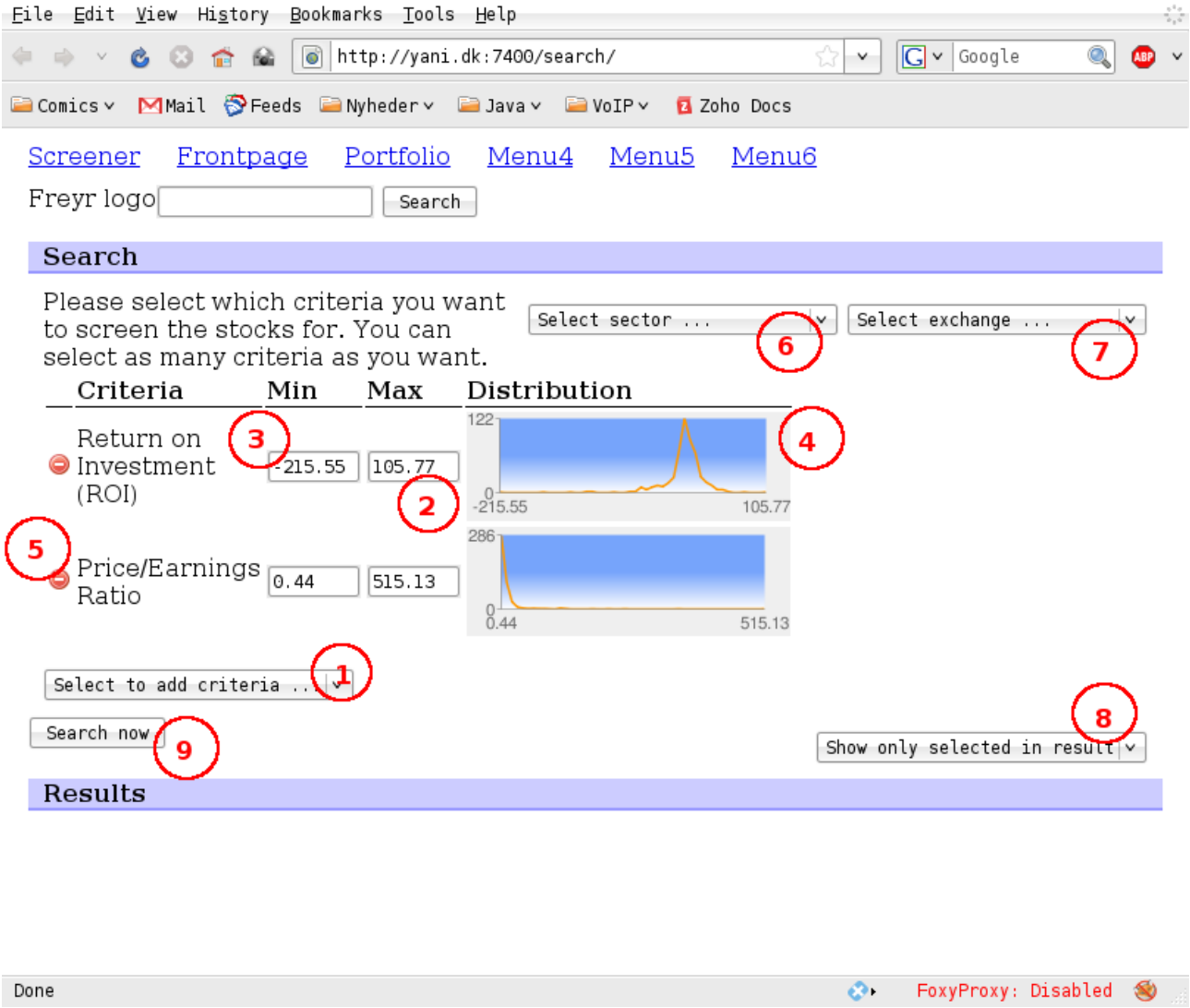
USER AND ADMIN GUIDE

Appendix C contains documentation for users and administrators on how they should use the program.

Users guide for Stock Screener v1.0

Searching

The search interface is reached by entering the /search/ URL. When the page is loaded, it is ready to use. Below is a screenshot of the search page with some comments.

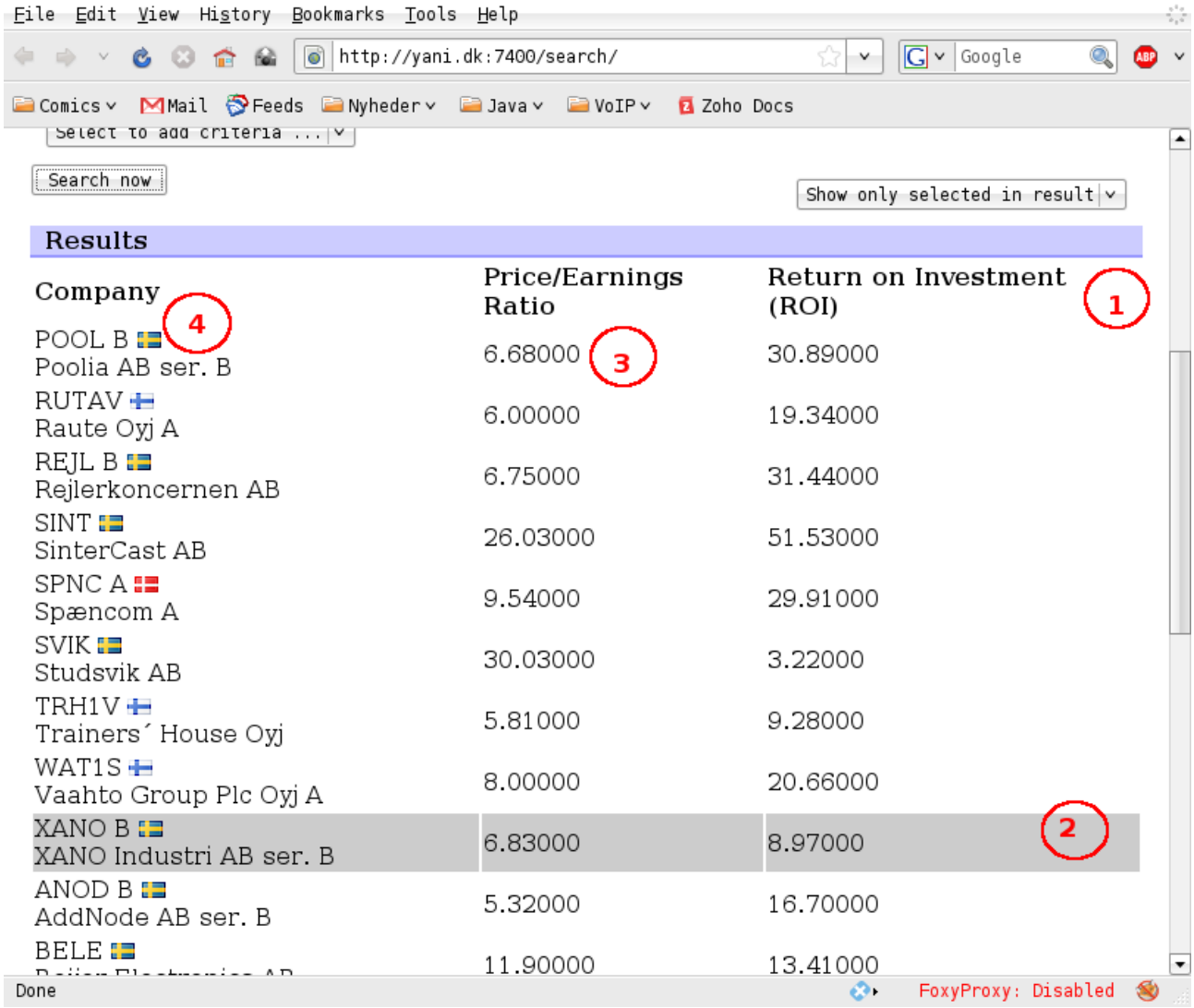


1) Here you can add criterias. By selecting a criteria in the dropdown, it will be added to the list of criterias above. When you enter the search page, the list is empty. It is possible not to enter any criteria.
2) For each criteria, it is possible to enter a maximum value for the criteria. You can also leave the field blank.
3) For each criteria, it is also possible to enter a minimum value, this field can also be left blank.
4) The graph shows the distribution of companies. X-axis shows the values, and Y-axis shows how many companies are in this range. For example ROI here, shows that all companies have a ROI between -255 and 105, and there are most companies with a ROI around 0.
5) You can click on the red stop button to remove a criteria if you do not wish to include it in your query anyway.
6) You can restrict the companies you are interested in to a specific sector.

7) You can also restrict the companies to a specific stock exchange.
8) It is possible to specify what to show in the result. If you want to show selected criteria, or all criteria available.
9) Finally, you click on the "Search now" button to get the results.

Viewing results

After clicking on "Search now" results are shown below the search form (if form is filled correctly and any companies are matching the query).



1) On top you can see the labels for each of the values, for example here P/E Ratio and ROI.
2) Each company is represented with one row in the result table. When you put mouse over some values, it will highlight the row.
3) The values are displayed inside the table, like how you read a normal table.
4) Each company is represented by its symbol, name, and a small flag indicating which country the company belongs to (actually on which stock exchange it is traded).

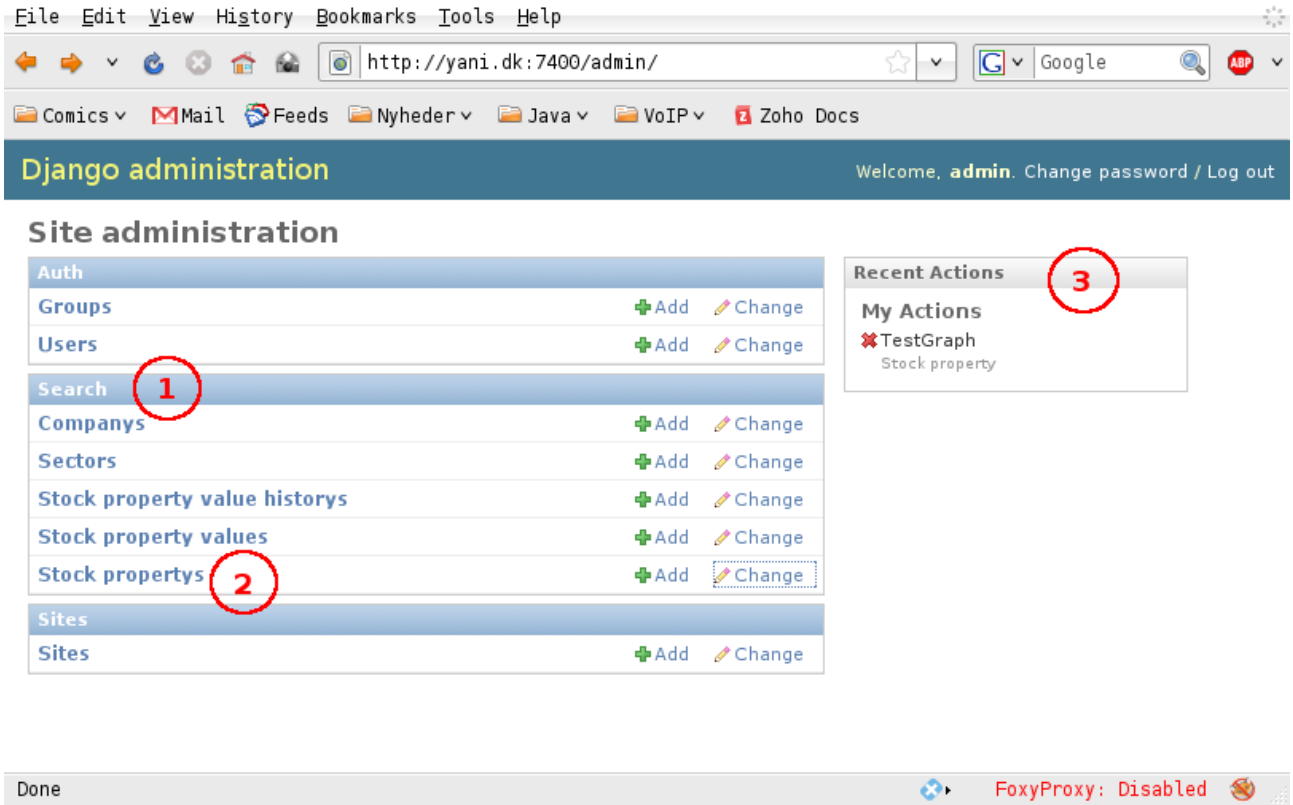
Administrators guide for Stock Screener v1.0

The administrator has different tasks he/she can do. In this document these will be described.

Task	Description
Maintain core data	To maintain core data means to be able to maintain the data stored in the database. The most important task here, is to add/remove stock properties.
Update list of companies	The list of companies interesting can change over time. Freyr has a web-service that contains interesting companies. This means to update the list of companies in the local database according to what is “true” from Freyr.
Update extracted values	This means to update the extracted values from the web. To load new – current – data into the database.

Maintaining core data

Maintaining core data is done using the administrator interface. The administrator interface can be reached using the /admin/ URL. Username and password is required. Once login has been completed, all data can be maintained.



1) Search is the application for Stock Screener. The list below is the models available for this.
2) Stock Propertys is the table that contains properties for extraction.
3) Recent actions can be seen by others, and to keep track of what happened.

For more information, please refer to the Django documentation on how to work with admin interface.

Updating list of companies

The list of companies can be update by using the manage.py command, as follows.

```
File  Edit  View  Terminal  Go  Help
kereena@sip1:~/stockscreener$ python manage.py updatesymbols
All OK ;-)
kereena@sip1:~/stockscreener$
```

1) Run the command: python manage.py updatesymbols
2) Check that you get the "All OK" message.
3) Possibly, verify in the admin interface that new symbols are available.

Updating extracted values

A new set of extracted values can be created by using the manage.py command, as follows.

```
File  Edit  View  Terminal  Go  Help
kereena@sip1:~/stockscreener$ python manage.py importall 1
Symbols=682, Properties=11
Extracting Price/Earnings Ratio for ALIV SDB -> None
Extracting Shares out for ALIV SDB -> None
Extracting Div. and yield for ALIV SDB -> None
Extracting Price/Sales Ratio for ALIV SDB -> None
Extracting Return on Investment (ROI) for ALIV SDB -> None
Extracting Return on equity (ROE) for ALIV SDB -> None
Extracting Total Revenue for ALIV SDB -> None
Extracting Net Income for ALIV SDB -> None
Extracting Sales 5yr Growth Rate for ALIV SDB -> None
Extracting EPS 5yr Growth Rate for ALIV SDB -> None
Extracting Gross Margin for ALIV SDB -> None
kereena@sip1:~/stockscreener$
```

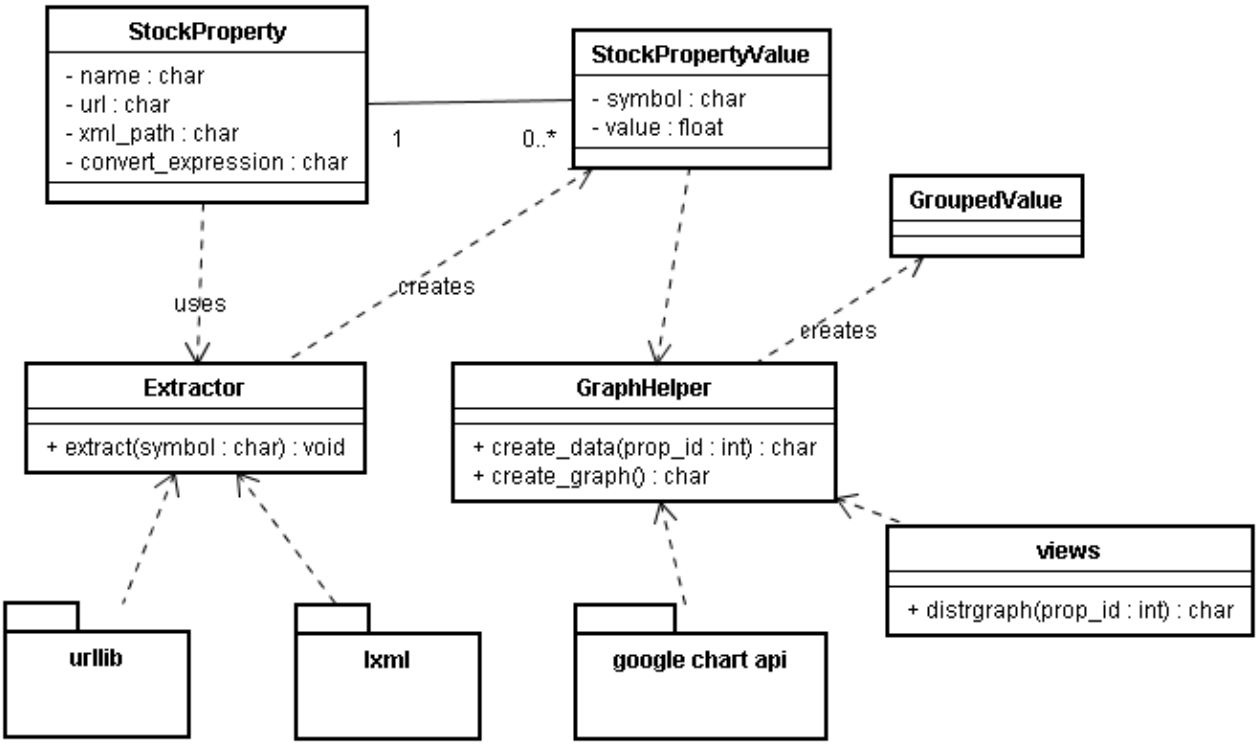
1) Run the command: python manage.py importall <number of companies to import>
    1. For example use 9999 if you want to import all.
2) Note that this will run for a very long time, to extract all values for all companies.

APPENDIX D

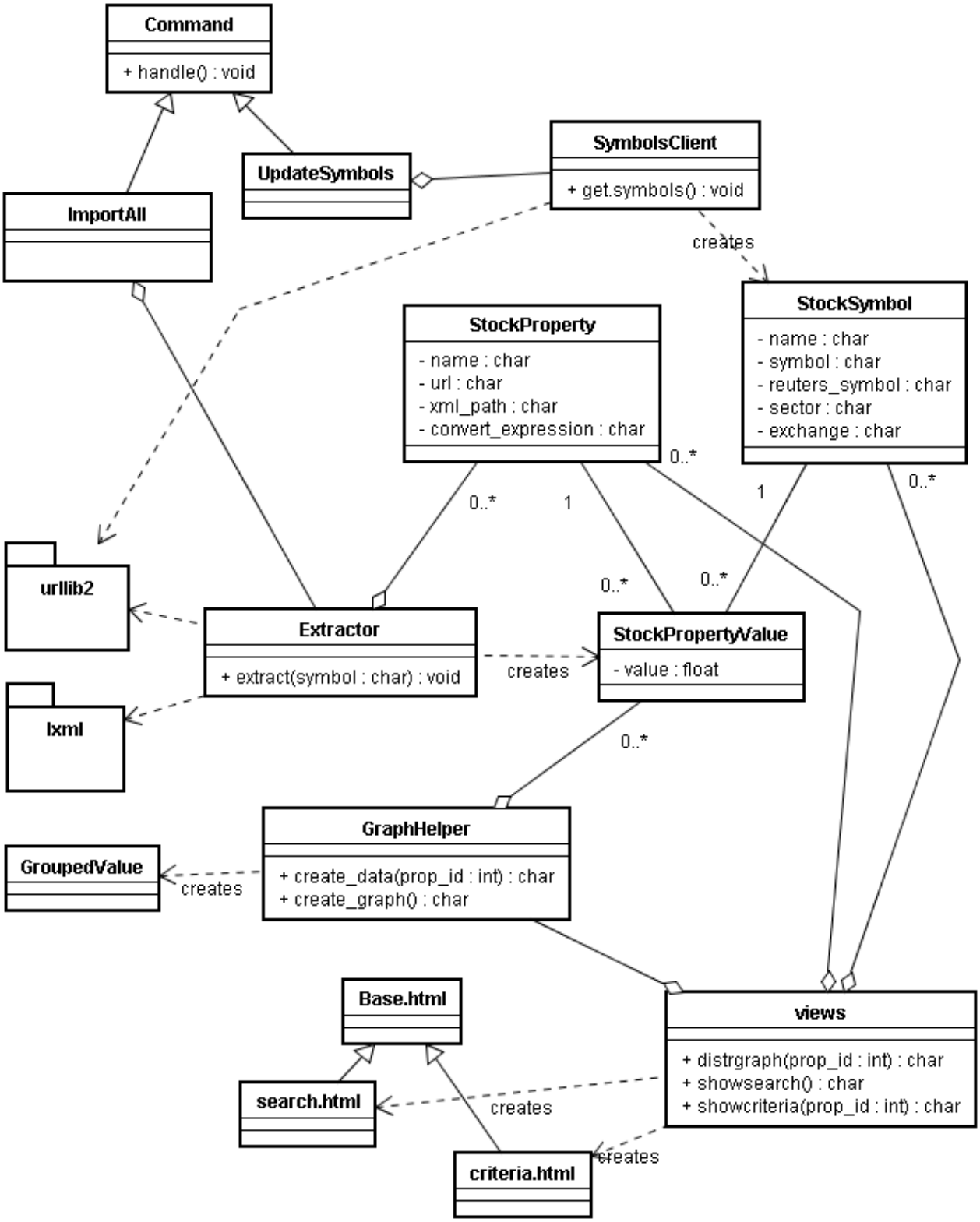
ITERATION DESIGN DOCUMENTATION

This appendix contains the intermediate output from the design process. For each iteration there are some UML documents, and by this it is possible to see the progress.

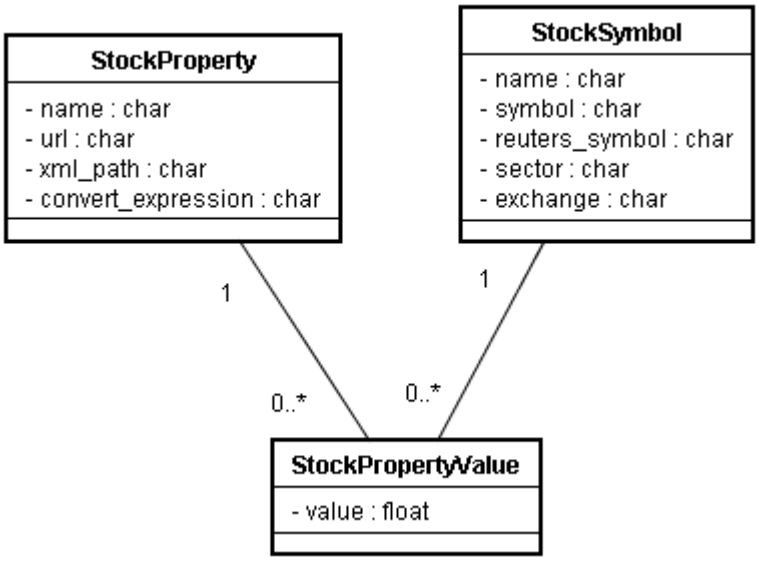
Iteration 1 UML class diagram



Iteration 2 UML class diagram

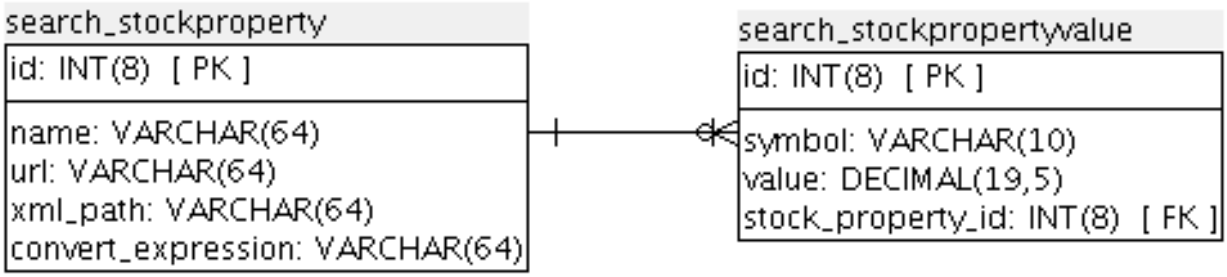


Iteration 2 UML business object model



Iteration 1 database E/R diagram

The diagram was created with another tool, before I found how to use MySQL workbench for creating the E/R.



APPENDIX E

RUNNING DEMONSTRATION

The CD contains a demonstration for Windows XP, this explains how to run it.

- Insert the CD.
- Copy the folder “demo” to somewhere on a write-able disk (hard-drive).
- Open the copy of “demo”.
- Double-click on the RunDemo.bat file.
- Two windows should open up:
  - DOS-window: Here the server is running.
  - Browser window: Read the instructions there. Click on the links to the demo server.

Note: The demo server uses 127.0.0.1 port 8000. Please make sure that this port is not already in use on your PC.

Note 2: The database in use is SQLite, not MySQL (which is production). All should work, but some queries can maybe work little different. This has not been tested.

APPENDIX F

REFERENCES TO EXTERNAL LIBRARIES

In my project I used a set of libraries and tools provided by others. The help of these were great, I did not have to invent everything from the bottom. Below is a list of the libraries and tools that I used, the links where more can be found, and a short description.

Name / link	Description
MySQL (5.1) http://dev.mysql.com/	A full database system. I used it for the database system in my project to store all data.
MySQL Workbench (5.0) http://dev.mysql.com/workbench/	A database design tool, which can also reverse-engineer database layout into E/R – which is what I used it for in this project.
Python (2.5) http://www.python.org/	A programming language. I used it in my project for all implementation.
Django (1.0) http://www.djangoproject.com/	A web application framework for Python. I used it in my project for all the parts related to the web-application, and ORM for the database.
Lxml (2.1) http://codespeak.net/lxml/	A XML library for working with XML in Python. I used it for XPath expressions, and parsing XML from Freyr webservices.
Prototype.js (1.6) http://prototypejs.org/	A Javascript library for AJAX web-applications. I used it for creating dynamic web-page.
BeautifulSoup (3.0) http://www.crummy.com/software/BeautifulSoup/	A library for working with HTML in Python. I used it for extracting information from web pages.
Google Chart API (1.0) http://code.google.com/apis/chart/	An online-library for creating charts. I used it for the distribution graph rendering.
COCOMO II Cost estimator http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html	COCOMO II is a online tool to calculate cost. I used it for cost estimation. (Note: Does only work with Internet Explorer)

APPENDIX G

XP WORKING PAPERS

I tried to use XP as much as possible in the process. In this appendix all my working papers can be found. These includes the User stories, and the Task Cards. While in the report I only present the final outcome of the design process, it is possible to see the intermediate steps by these working papers.

APPENDIX H

COST ESTIMATION OUTPUT

Here we can see the output of working with the COCOMO II cost estimation tool.

Project Analogy Parameter: Total Effort in Person-Months
Project Baseline Value: 3 (Person-Months)
Current Project Labor Rate: 20000 (Dollars / Person-Months)
Project Calibration Source: COCOMO Default Values

#	Basis Type	Estimation Basis		Old Rating Level	New Rating Level	Estimated Cost	Estimated Effort (In PM)
0		(starting)				$60000.00	3
1	Cost Driver	Product-	Required Software Reliability	VH (1.26)	N (1.00)	$47619.05	2.38
2	Cost Driver	Product-	Database Size	N (1.00)	H (1.14)	$54285.71	2.71
3	Cost Driver	Product-	Product Complexity	L (0.87)	H (1.17)	$73004.93	3.65
4	Cost Driver	Product-	Developed for Reusability	L (0.95)	N (1.00)	$76847.29	3.84
5	Cost Driver	Platform-	Execution Time Constraint	XH (1.63)	H (1.11)	$52331.59	2.62
6	Cost Driver	Personnel-	Analyst Capability	N (1.00)	H (0.85)	$44481.85	2.22
7	Cost Driver	Personnel-	Programmer Capability	H (0.88)	N (1.00)	$50547.56	2.53
8	Cost Driver	Personnel-	Application Experience	L (1.10)	VL (1.22)	$56061.84	2.8
9	Cost Driver	Personnel-	Platform Experience	N (1.00)	H (0.91)	$51016.27	2.55
10	Cost Driver	Personnel-	Language and Tool Experience	L (1.09)	VL (1.20)	$56164.70	2.81
11	Cost Driver	Project-	Schedule	N (1.00)	H (1.00)	$56164.70	2.81
Total						$56164.70	2.81

VH = Very High, H = High, XH = Extra High, N= Nominal, L = Low, VL = Very Low