University of Nevada, Reno

New Money Data Vending

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Nolan Perry Burfield

Dr. Frederick C. Harris, Jr., Thesis Advisor

December, 2016

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

NOLAN PERRY BURFIELD

Entitled

New Money Data Vending

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Advisor

Dr. Sergiu M. Dascalu, Committee Member

Dr. Gregory R. Stone, Graduate School Representative

David W. Zeh, Ph.D., Dean, Graduate School

December, 2016

Abstract

This thesis presents a system, New Money Data Vending, that will provide users with information on the stock market. This information is to be purposed to executing trades to maximize profit. The system is broken down into three parts. The first part is the system that pulls that data on the financial system. The purpose of this thesis discusses the pulling of market data, historical prices of a stock, and financial data, company filings with the Securities and Exchange Commission. This part is built to automate the process and requires no user interaction to properly build the database of the financial data. The second part is a RESTful service that will provide access to the database. The rest service returns JSON objects of the queried financial or market data. The final part is a user interactive website in order to present the users with the data stored in the database, and allow for the storage of this data.

Dedication

I dedicate this thesis to my family and friends who have supported me.

Acknowledgments

I would like to thank my adviser, Dr. Frederick Harris, and my committee members Dr. Sergiu Dascalu and Dr. Gregory Stone for their time and suggestions. I would like to thank Raja Singh for the initial work on the system that lead to the thesis. Of course, I would like to thank my family for their support.

This material is based in part upon work supported by the National Science Foundation under grant number(s) IIA-1329469. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Market capitalization of the United States is larger than 19 trillion dollars [46], and within the United States 54 percent of Americans are invested in the stock market [10]. Individuals that are trading in the markets do so to provide themselves with retirement, or make large purchases down the road. With the advancement of technology the average stock trader has access to their own at home trading platform to execute trades at as low as $5 a transaction [25]. With the ability to generate a steady income from the markets there is also the ability to lose in the markets. Banking companies are required to guarantee the safety of clients money (up to $250,000 [9]), but the securities market has no such guarantee. The Securities and Exchange Commission (SEC) recommends conducting research about the company prior to purchasing stares of their company [26]. Information on the securities market is difficult to find; large companies dominate the supply of the data. The Bloomberg Terminal, which has subscription costs in the thousands [33], is one such company that capitalizes on the need for market and financial data to drive the securities trading companies.

This thesis presents a system that automatically pulls market data and stores this information into a database to be provided to the average user in a fashion that most users can understand. This is referring to the data providers that do exist currently do not provide a simple, user friendly method of data collection that can be local to all users. Market data is pulled from Yahoo stocks, which is the daily price points of stocks. Financial data is gathered from an SEC database named EDGAR. The EDGAR database is the SEC stored filings from all publicly traded companies.

Since the SEC has a goal of transparency between investors and companies they requires public companies to disclose "meaningful financial and other information to the public." [32] The form focused on in this thesis is the Form 4 insider trading form (one of the most filed forms in the EDGAR database). The automation of the gathering and storage of this data keeps the system light and minimal overhead minus the need to keep the system running at speeds reliable to the users. The user of this system will be given a user interface of the data stored in the database, and options to view this information on each company.

The rest of this thesis is structured as follows: a little background into the financial markets, the libraries used, and some related work is presented in Chapter 2. Chapter 3 goes over how the system is designed with use cases, requirements, and the articheture of the system. Chapter 4 gives the build of each of the four pieces of this system; the resource builder, the RESTful service, and the website. Finally the paper wraps up in Chapter 5 with a discussion of the future work and the conclusion.

Chapter 2

Background and Related Work

2.1 Market Data

Market data is easily available to the average person, but in a format that does not allow analysis to be performed on it. This data comes in the forms of charts from many different companies (eg. Yahoo, Google, Scottrade, TD Ameritrade, etc.). Market data is the historical prices of a traded commodity; in this case the commodity is stocks. These prices include the opening cost, closing cost, highest cost in the day, lowest cost in the day, and volume. There is a limited number of published papers on the means an everyday trader could pull this market data. Chan [5] mentions many databases to gather this information from, but does not provide a method of extraction. Most of there databases are not free to access except Yahoo, that is why the focus of the market data comes from Yahoo Finance. The Yahoo database requires a programming background in order to extract the market data in the large scale means of analytics. Hilpisch [13] provides the code in Python to achieve this and [8] and [49] give tutorials to write the code, however this is not the ideal method for an everyday user to get market data.

The Yahoo stocks data provides a comprehensive list of data to the user. This data is stored in a database that is open to query that will return data in a JSON or CSV format to the user. This is a free method of data extraction and storage to fill the New Money Data Vending database with all previous market data. Using Yahoo will be the method (discussed later in Chapter 4) to extract market data for

the purpose of this application [49].

NASDAQ offers a service to pull market data as well, but this comes at a cost to the user. The fees for pulling data is not cheap, NASDAQ Data-on-Demand has an annual fee of $2,400. This price is not sustainable for an average trader [19].

2.2 Financial Data

2.2.1 Overview

Financial data is a crucial part of a companies success. The information provided gives insight into revenue, debt, company owners, and futures of the company. The vending of this data is largely corporate, so publications on this are not available. A few were found discussing the crawling of data associated to EDGAR (EDGAR is discussed more in Section 2.2.2). Garcia and Norli [11] discuss crawling EDGAR for the Form 8K, which is filed by companies when giving a public announcement. The 8K form is the second highest filed form behind the Form 4. Their goal in gathering the Form 8K information was to find CEO turnovers, and they achieved this using Pearl to pull and parse the files. The other paper by Lyon [18] goes over the gathering of central index keys and correlating them to the company in a graphical user interface. The financial market is large and secretive, and that is one big reason for creating the New Money Data Vending application.

2.2.2 Electronic Data Gather and Retrieval

The Electronic Data Gather and Retrieval (EDGAR) database is the SEC storage center for all the filings from the publicly traded companies. This is not limited to the Form 4 documents that are discussed in this thesis. EDGAR has been around since 1994 and has more than 4 million filings in the database. EDGAR uses central index keys (CIK) to correlate files to companies. The EDGAR database has an FTP access point, and also access to the files through the SEC website. The SEC site has a graphical user interface that is difficult to navigate, and document organization is

not well. Providing a different infrastructure to view these forms in a cleaner method is more beneficial to the users [31].

2.2.3 Xignite

Xignite is one company that rivals with the data vending industry, including the largely dominate Bloomberg [47]. This company provides an API to subscribers to pull market and financial data. The company however does not provide inside into how they disseminate their financial data. Xignite is well developed and offers subscriptions to many different markets including foreign. The price of this product was based off of an annual subscription, but Xignite claims price is based off "'how' and 'the scale' the data will be used or displayed" [48].

2.2.4 Form 4

Overview

The Form 4 for SEC filings is an Insider Trading Form; the Form 4 is only one of the insider forms, there are also Form 3 and Form 5. The context of this insider trading is the buying and selling of stock of a company that a person is categorized as an insider. The SEC requires insiders to file with them after conducting a trade (Figure 2.1 is an example of a Form 4 document). Those that are considered insider are people that are corporate insiders (company officers or directors), or those that own ten percent or more of the company. These people are required to file a "statement of ownership regarding those securities" using Form 3. Form 5 is then used to provide more transactions that did not show up on the Form 4 documents. There are also amendments to these forms and they are the number followed by an A (ex. Form 4/A). These forms provide valuable information to the securities market, and papers discussing the parsing of these documents were not able to be found [28].

SEC Form 4

FORM 4

UNITED STATES SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

STATEMENT OF CHANGES IN BENEFICIAL OWNERSHIP

Filed pursuant to Section 16(a) of the Securities Exchange Act of 1934 or Section 30(h) of the Investment Company Act of 1940

OMB APPROVAL	
OMB Number:	3235-0287
Estimated average burden hours per response:	0.5

☐ Check this box if no longer subject to Section 16. Form 4 or Form 5 obligations may continue. *See* Instruction 1(b).

1. Name and Address of Reporting Person*	2. Issuer Name **and** Ticker or Trading Symbol	5. Relationship of Reporting Person(s) to Issuer (Check all applicable)
BERKSHIRE HATHAWAY INC	MOODYS CORP /DE/ [MCO]	Director X 10% Owner
(Last) (First) (Middle)	3. Date of Earliest Transaction (Month/Day/Year)	Officer (give title below) Other (specify below)
3555 FARNAM STREET	10/28/2009	
(Street)	4. If Amendment, Date of Original Filed (Month/Day/Year)	6. Individual or Joint/Group Filing (Check Applicable Line)
OMAHA NE 68131		Form filed by One Reporting Person
(City) (State) (Zip)		X Form filed by More than One Reporting Person

Table I - Non-Derivative Securities Acquired, Disposed of, or Beneficially Owned

1. Title of Security (Instr. 3)	2. Transaction Date (Month/Day/Year)	2A. Deemed Execution Date, if any (Month/Day/Year)	3. Transaction Code (Instr. 8)		4. Securities Acquired (A) or Disposed Of (D) (Instr. 3, 4 and 5)			5. Amount of Securities Beneficially Owned Following Reported Transaction(s) (Instr. 3 and 4)	6. Ownership Form: Direct (D) or Indirect (I) (Instr. 4)	7. Nature of Indirect Beneficial Ownership (Instr. 4)
			Code	V	Amount	(A) or (D)	Price			
Common Stock	10/28/2009		S		1,133,027	D	$24.8637(1)	38,086,285	I	See footnotes 3 and 4.(3)(4)
Common Stock	10/29/2009		S		19,600	D	$25.2728(2)	38,066,685	I	See footnotes 3 and 4.(3)(4)

Table II - Derivative Securities Acquired, Disposed of, or Beneficially Owned (e.g., puts, calls, warrants, options, convertible securities)

1. Title of Derivative Security (Instr. 3)	2. Conversion or Exercise Price of Derivative Security	3. Transaction Date (Month/Day/Year)	3A. Deemed Execution Date, if any (Month/Day/Year)	4. Transaction Code (Instr. 8)		5. Number of Derivative Securities Acquired (A) or Disposed of (D) (Instr. 3, 4 and 5)		6. Date Exercisable and Expiration Date (Month/Day/Year)		7. Title and Amount of Securities Underlying Derivative Security (Instr. 3 and 4)		8. Price of Derivative Security (Instr. 5)	9. Number of derivative Securities Beneficially Owned Following Reported Transaction(s) (Instr. 4)	10. Ownership Form: Direct (D) or Indirect (I) (Instr. 4)	11. Nature of Indirect Beneficial Ownership (Instr. 4)
				Code	V	(A)	(D)	Date Exercisable	Expiration Date	Title	Amount or Number of Shares				

Figure 2.1: A top half of the SEC rendition of a Form 4 document

Uses

The Form 4 filings are useful to trades in order to see movements in trends that will be in the companies future. The transactions on these forms come as a type either derivative or non-derivative. The non-derivative is a relationship between the actual share of the company being traded. Derivative transactions come in the form of options, which have an underlying security value. There are theories that apply to reading the trends of insider action, since insiders have knowledge of company futures that the average person may not see. The New Money Data Vending provides the data of the insider actions, their relation to the company, and the quantity that is being traded upon.

2.3 Libraries and Frameworks

The three parts of the New Money Data Vending depend on multiple libraries and frameworks. This includes integrated development environment (IDE) eclipse, services for networking, and user interface styling. These libraries and frameworks are listed in this section.

Java 8 is the programming language that handles the networking and the database connection. Java is concurrent, class-based, and object oriented programming language. Java is highly regarded in the networking capabilities it has, and provides better speeds than other networking options. Java offers programmers many standard classes, which is a great choice for development [23].

MySQL offers a multi-threaded and multi-user structured query language (SQL) database. MySQL is designed to handle heavy loaded systems. The data is stored in structured tables, which can be accessed via multiple threads. The MySQL database was chosen for its robustness, easy of setup, and common language (SQL) [22].

The Spring Framework is a Java platform that offers the infrastructure to handle Java applications. The Spring Framework has libraries that will connect to the database and the model-view-controller networking structure. Spring provides dependency injection to alleviate the need to dependencies being strongly coupled in an application. Dependency injection allows for interfaces of objects, therefore changing an object from one type to the other will have no problems with the system (ex. switching from a MySQL database to MongoDB). Spring has a Core container that includes the fundamentals of Spring with the Beans for dependency injection, and the context which inherits the Beans. The next level handles the integration to the database and the networking of the application. This level is the Alliance-compliant aspect-oriented programming implementation (the key part to the decoupling), messaging, and data access/integration [24].

Spring Boot offers a Spring build platform that comes pre-built to "Just Run" as a web application. Spring Boot runs the Spring application as a Java Jar or with

a Spring runner. Spring Boot uses Maven to handle packages, which is useful to port the same application as it to another machine with minimal alterations to be provided to the machine (not to the application itself). The use of this software on the New Money Data Vending software was to get a working skeleton with proper packages up and running quickly [45].

Apache Maven handles the project construction and libraries through a POM file. Maven allows the project to be built and run across machines. The Maven software will even handle reporting and documentation [2].

Apache Commons Net handles the client side of networking for application. The use for Apache Commons Net in New Money Data Vending is to run the file transfer protocol connections to the EDGAR database and extract the files. This library will allow for the anonymous authentication as well to the Securities and Exchange Commission EDGAR server. [1].

The Java Database Connectivity (JDBC) provides a library to connect to a MySQL database through Java. This connector is not explicitly called through the New Money Data Vending application, but is the backbone of the connection the Spring utilizes. The connection allows for Spring to POST and GET data from the MySQL database used [20].

Hyper Text Markup Language (HTML) is the language that a web browser reads to render the display. HTML uses tags to describe to the web browser on how exactly to display the content. HTML is used in this application to design the web page for the user interface [42].

Cascading Style Sheets (CSS) is used to describe how to style HTML content. CSS is used to define layout, design, and variations (ex. The display for screen resolutions). This is used in the application to add style to the user interface and not have a standard display that would be uninteresting to the user [41].

Bootstrap is a front-end framework making the design of web pages simplistic and consistent across devices. Bootstrap is a HTML, CSS, and Javascript framework to make the web page responsive. The application utilizes Bootstrap in order to make

nice features for the user and prevent reworking a project front-end to work for mobile devices [4].

Javascript allows for dynamic web page content. This library is used in the New Money Data Vending application through the use of Bootstrap and the TechanJS charts. Javascript is a necessary part of this application in order to do the data visualization that is present to the user, and the data handing to the front-end [43].

TechanJS is the stock charts that are used for the data visualization of the market data. This library is written in CSS and Javascript. These charts available are built with market data in mind, and have a pre-built candlestick feature to display the particular market data that is collected by New Money Data Vending [6].

Eclipse is an integrated development environment (IDE) specifically for Java applications. The use of this is invaluable since the IDE provides error checking, code completion, package management, automatic imports, and the ability to run the Java application from inside the IDE. Eclipse also provides plugins to run the Spring Boot and builds Maven projects [7].

Chapter 3

Design

3.1 Overview

New Money Data Vending has three parts associated to it: resource builder, rest service, and the web client. These three parts are all key to providing the end user with the market and financial data that is asked for. In this section the requirements, use cases, and architecture of the New Money Data Vending system is discussed.

3.2 New Money Data Vending Requirements

3.2.1 Functional Requirements

The functional requirements of the three parts of New Money Data Vending are combined into one single table. The functional requirements can be found in Table 3.1.

3.2.2 Non-functional Requirements

The non-functional requirements of the three parts of New Money Data Vending are combined into one single table. The non-functional requirements can be found in Table 3.2.

Table 3.1: The functional requirements of the New Money Data Vending

Number	Description
FR01	The website shall show a list of companies stored in the database
FR02	The website shall show information on a single company
FR03	The website shall display a single company raw historical data
FR04	The website shall display a single company graph of the historical data
FR05	The website shall show a single company Form 4 financial filings
FR06	The website shall show information on a single insider
FR07	The website shall show the filings associated to a single insider
FR08	The REST service shall send a list of all companies
FR09	The REST service shall send a companies historical data
FR10	The REST service shall send company data
FR11	The REST service shall send company insider transactions
FR12	The REST service shall send insider data
FR13	The REST service shall send all data on a single insider
FR14	The Resource builder shall get company market data from Yahoo
FR15	The Resource builder shall get Form 4 information from EDGAR
FR16	The Resource builder shall parse Form 4
FR15	The Resource builder shall store data in the database

Table 3.2: The Non-functional requirements of the New Money Data Vending

Number	Description
NFR01	The website will interface with New Money Rest
NFR02	The website will be written with Java Spring
NFR03	The REST service will interface with MySQL
NFR04	The REST service will be written with Java Spring
NFR05	The Resource Builder will interface with MySQL
NFR06	The Resource Builder will be written with Java

3.3 Use Case Modeling

3.3.1 Overview

The use cases of the New Money Data Vending system will be split into the three components that represent it. The website will focus on the user as the actor; interactions with the front end. The website will be making the calls for data to the REST service. The web client will be the next actor interacting with the RESTful services provided. The last actor in the use cases of the system will be the Resource Builder actor. This is the one that will have the calls to build the database with the proper market and financial data.

3.3.2 Detailed Use Cases

The Sections 3.3.3, 3.3.4, and 3.3.5 define the three section of the New Money Application use cases.

3.3.3 New Money Resource Builder Use Cases

A detailed use case diagram of New Money Resource Builder can be found in Figure 3.1.

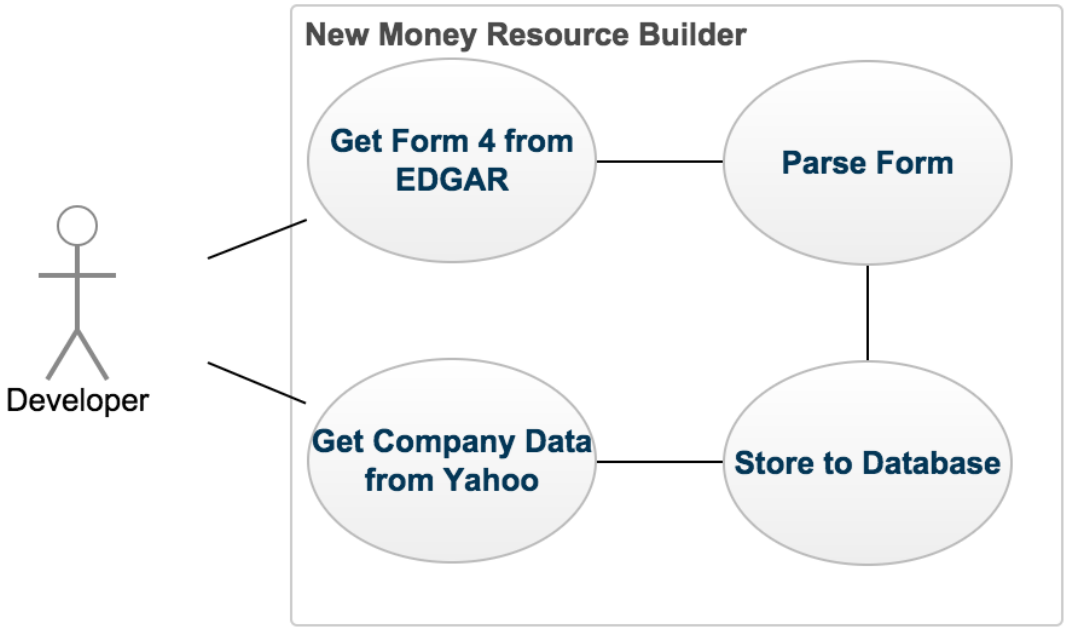


Figure 3.1: This diagram represents the relationship between the developer actor and the New Money Resource Builder

Get Company Information from Yahoo

This will pull the historical data from a selected company from the Yahoo Finance service. It will also parse this data and store it in the database.

Get Form 4 from EDGAR

This will pull all the Form 4 documents associated to a certain company.

Parse Form 4 Document

This will run the parse on the pulled Form 4 document, and store the information in the database.

3.3.4 New Money RESTful Service Use Cases

A detailed use case diagram of New Money RESTful service can be found in Figure 3.2.

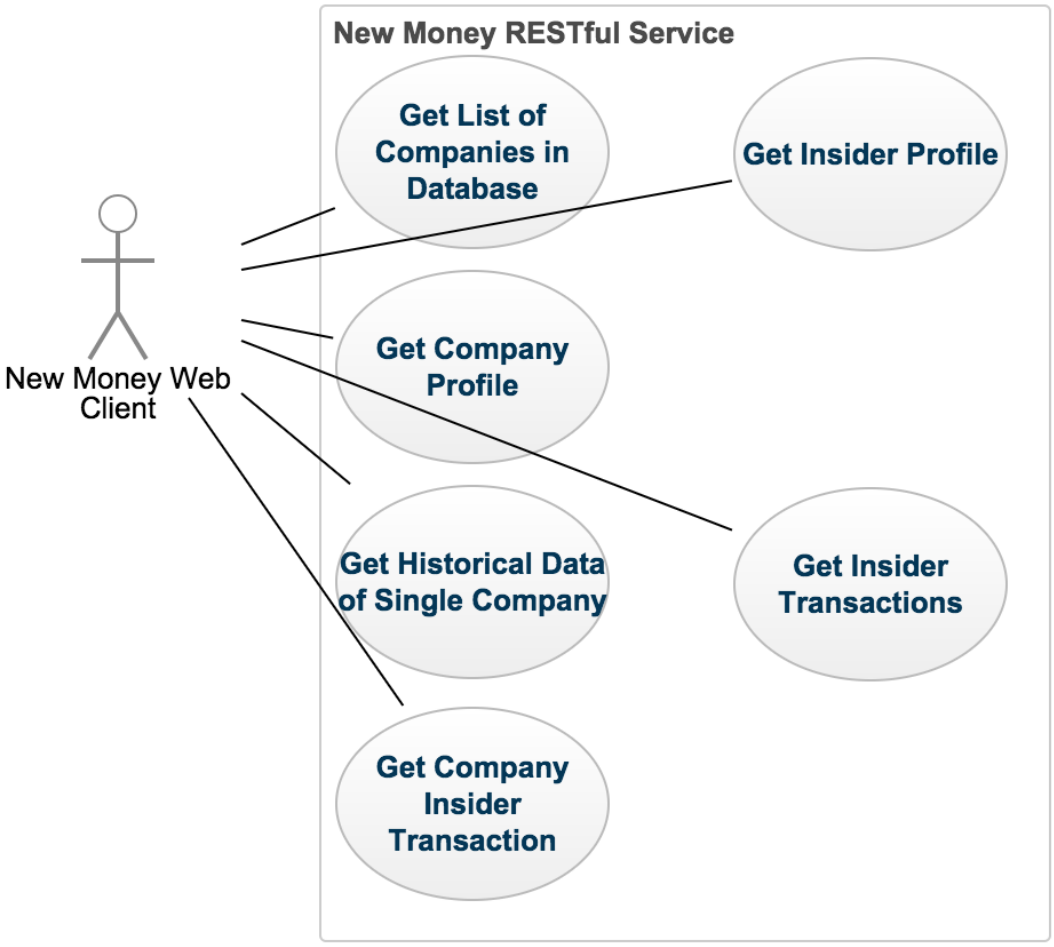


Figure 3.2: This diagram represents the relationship between the New Money Web Client actor and the New Money Rest Service

Get a List of Companies in the Database

This will return a list of all the companies that are in the database of New Money Data Vending.

Get the Historical Data of a Single Company

This will return a list of all the historical prices of a company.

Get Company Profile

This will return information associated to a company. This is the name, ticker, cik, and address.

Get Company Insider Transactions

This will return all the insider transactions of a single company.

Get Insider Profile

This will return the information associated to an insider. This is the name, cik, and address.

Get Insider Transactions

This will return a list of all the insider transactions for all companies.

3.3.5 New Money Website Use Cases

A detailed use case diagram of New Money Website can be found in Figure 3.3.

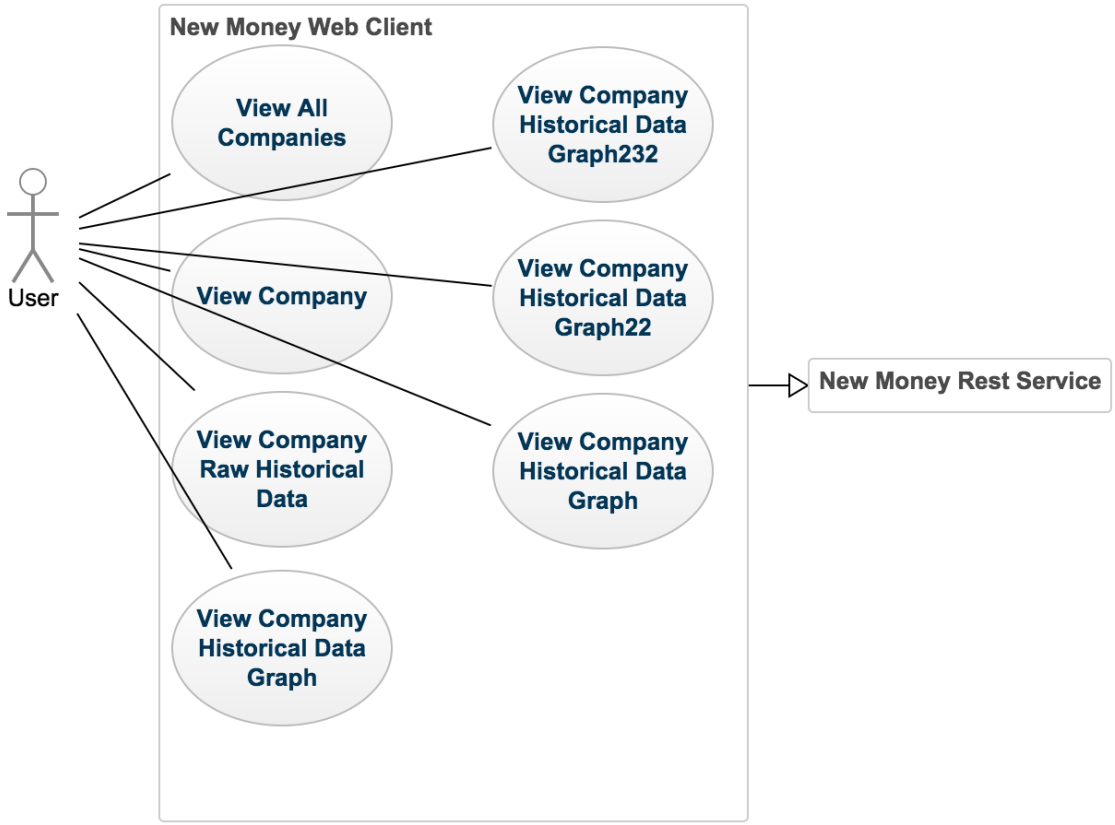


Figure 3.3: This diagram represents the relationship between the user actor and the New Money Web Client

View All Companies

This will show all the companies that are stored in the database for viewing.

View Company

This will allow the user to view the information on a single company.

View Company Raw Historical Data

This will allow the user to view the raw historical data stored in the database in a table format in the web browser.

View Company Historical Data as Graph

This will allow the user to view the historical data stored in the database in an intractable graph.

View the Insider Transactions on a Company

This will show the user the selected companies insider transactions. These are the purchases and sales related to that company specifically.

View an Insider Profile

This will allow the user to view the profile associated to a single insider.

View the Transactions for a Single Insider

This will allow the user to view the profile of a single insider. This is all the trades that they executed across all companies.

3.4 Database

The New Money Data Vending stores all the market and financial data in a MySQL database. The SQL database requires the definitions of tables, and the types of values that are stored in those tables. The tables in the New Money Data Vending data base are as follows:

- Companies table will store the information on a single company (Figure 3.4)
- Insiders table will store the information on a single insider (Figure 3.5)
- Historical Data table will store each record of a company market data throughout the companies history (Figure 3.6)
- Form 4 Files table will store the header information on a Form 4 document (Figure 3.7)

- Non-derivative table stores the data in the non-derivative tables on the Form 4 document (Figure 3.8)
- Derivative table stores the data in the derivative tables on the Form 4 document (Figure 3.9)

companies		
Primary	cik	VARCHAR(30)
	name	VARCHAR(255)
	ticker	VARCHAR(30)
	exchange	VARCHAR(30)
	street1	VARCHAR(255)
	street2	VARCHAR(255)
	city	VARCHAR(30)
	state	VARCHAR(30)
	zip	VARCHAR(30)
	stateDescription	VARCHAR(4)

Figure 3.4: A zoomed portion of the companies table on the ERD

insiders		
Primary	cik	VARCHAR(30)
	name	VARCHAR(255)
	street1	VARCHAR(255)
	street2	VARCHAR(255)
	city	VARCHAR(30)
	state	VARCHAR(30)
	zip	VARCHAR(30)
	stateDescription	VARCHAR(4)

Figure 3.5: A zoomed portion of the insiders table on the ERD

historical_data		
Primary	company_cik	VARCHAR(30)
Primary	year	INT
Primary	month	INT
Primary	day	INT
	open	DOUBLE
	high	DOUBLE
	low	DOUBLE
	close	DOUBLE
	volume	DOUBLE
	adjusted_close	DOUBLE

Figure 3.6: A zoomed portion of the historical data table on the ERD

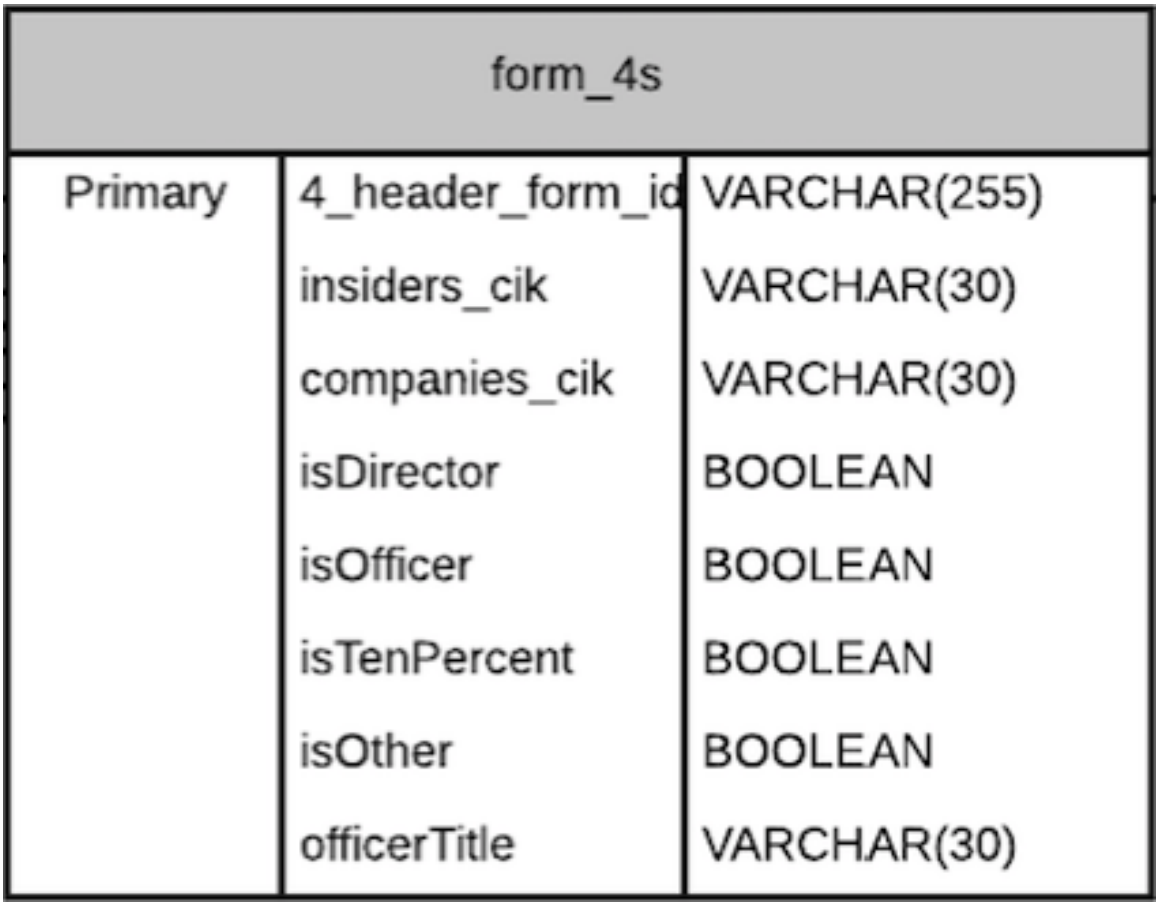


Figure 3.7: A zoomed portion of the Form 4 table on the ERD

nonderivative		
Primary	nonderivative_id	BIGINT
Primary	4_header_form_id	VARCHAR(255)
	security_title	VARCHAR(255)
	transaction_date	VARCHAR(30)
	deemed_execution_date	VARCHAR(30)
	transaction_form_type	CHAR
	transaction_code	CHAR
	equity_swap_involved	BOOLEAN
	transcation_timeliness	CHAR
	transaction_shares	DOUBLE
	transaction_price_per_share	DOUBLE
	transaction_aquired_disposed_code	CHAR
	shares_owned_following_transaction	DOUBLE
	value_owned_following_transaction	DOUBLE
	direct_or_indirect_ownership	CHAR
	nature_of_ownership	VARCHAR(255)
	security_title_footnote	BLOB
	transaction_date_footnote	BLOB
	deemed_execution_date_footnote	BLOB
	transaction_code_footnote	BLOB
	transcation_timeliness_footnote	BLOB
	transaction_shares_footnote	BLOB
	transaction_price_per_share_footnote	BLOB
	transaction_aquired_disposed_code_footnote	BLOB
	shares_owned_following_transaction_footnote	BLOB
	value_owned_following_transaction_footnote	BLOB
	direct_or_indirect_ownership_footnote	BLOB
	nature_of_ownership_footnote	BLOB

Figure 3.8: A zoomed portion of the non-derivative table on the ERD

derivative		
Primary	derivative_id	BIGINT
Primary	4_header_form_id	VARCHAR(255)
	security_title	VARCHAR(255)
	conversion_or_exercise_price	DOUBLE
	transaction_date	VARCHAR(30)
	deemed_execution_date	VARCHAR(30)
	transaction_form_type	CHAR
	transaction_code	CHAR
	equity_swap_involved	BOOLEAN
	transcation_timeliness	CHAR
	transaction_shares	DOUBLE
	transaction_total_value	DOUBLE
	transaction_price_per_share	DOUBLE
	transaction_aquired_disposed_code	CHAR
	exercise_date	VARCHAR(30)
	expiration_date	VARCHAR(30)
	underlying_security_title	VARCHAR(255)
	underlying_security_shares	DOUBLE
	underlying_security_value	DOUBLE
	shares_owned_following_transaction	DOUBLE
	value_owned_following_transaction	DOUBLE
	direct_or_indirect_ownership	CHAR
	nature_of_ownership	VARCHAR(255)
	security_title_footnote	BLOB
	conversion_or_exercise_price_footnote	BLOB
	transaction_date_footnote	BLOB
	deemed_execution_date_footnote	BLOB
	transaction_code_footnote	BLOB
	transcation_timeliness_footnote	BLOB
	transaction_shares_footnote	BLOB
	transaction_total_value_footnote	BLOB
	transaction_price_per_share_footnote	BLOB
	transaction_aquired_disposed_code_footnote	BLOB
	exercise_date_footnote	BLOB
	expiration_date_footnote	BLOB
	underlying_security_title_footnote	BLOB
	underlying_security_shares_footnote	BLOB
	underlying_security_value_footnote	BLOB
	shares_owned_following_transaction_footnote	BLOB
	value_owned_following_transaction_footnote	BLOB
	direct_or_indirect_ownership_footnote	BLOB
	nature_of_ownership_footnote	BLOB

Figure 3.9: A zoomed portion of the derivative table on the ERD

Specifics on what each table data represents will be in Section 4.1. The entity relationship diagram of the database explained above can be seen in Figure 3.10.

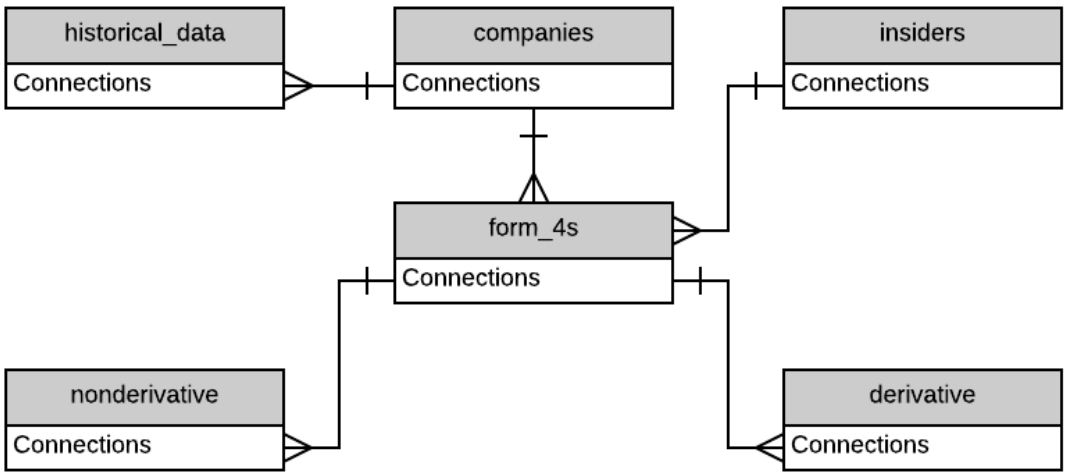


Figure 3.10: This is an entity relationship diagram of only table names of the New Money Data Vending database for the market and financial data

3.5 Architecture

As stated previously there are three parts to the New Money Data Vending system. Each of these parts contain their own type of architecture in their individual design, and then the interconnections of each rely on the connection to the parts they have. Below each of the three parts are described. Figure 3.11 is a visual of the architecture layout.

New Money Website Architecture

The website was built in Java Spring Boot and follows the basic model-view-controller style. The controller of the system makes the REST calls the the REST service. The data retrieved from this service is then placed into one of the Model definitions. These models define the data that is expected from the REST service in order to create consistency in the views of the data. The view will be the front end to the user, offering the user the ability to see the data of the model.

New Money RESTful Service Architecture

The RESTful service of the New Money system has a model and controller setup. The model defines the tables of the MySQL database, and handles the connection between getting and receiving that data. The controller of the system will handle the incoming RESTful requests for data. The controller gets the data from the database and packages the data in a JSON format and send it back.

New Money Resource Builder Architecture

The Resource Builder is also a model and controller setup. The models are associated to the REST models since the two share the same database. The controller of this system however is in charge of the connections to Yahoo and EDGAR and then the subsequent parsing of the data provided from those two services.

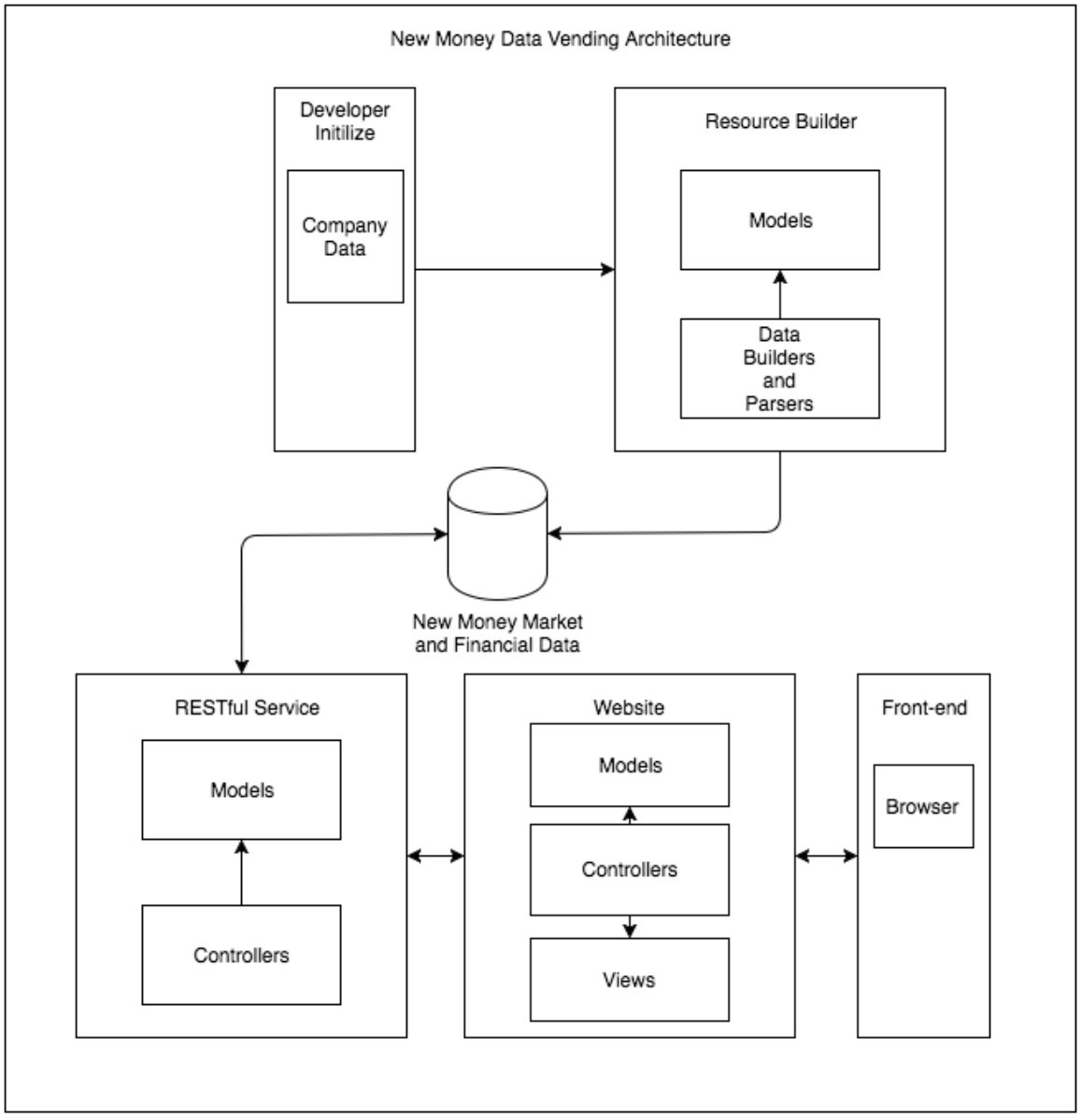


Figure 3.11: An architecture diagram of the New Money Data Vending system

Chapter 4

Implementation

4.1 New Money Resource Builder

4.1.1 Overview

The New Money Resource Builder is the system that builds the database and populates it with the data. This is built with Java and takes advantage of the Java Spring database interface, and the dependency injections. The interface with the Java Spring does a lot of the heavy lifting when the system is initialized. Spring checks that the database exists, the tables exist, and ensures that exceptions with the connections and saving of the data is handled. The system on startup will run a setup phase that will pull all the data of all the companies that is saved in a master list of companies built by the developers. This list is stored as a CSV and is parsed with the company information that is a part of the database. These values that all companies have are, and relate to, the database table companies:

- **Name** The name of the publicly traded company.
- **Ticker** The ticker associated to the company, used to reference the company on the exchange.
- **CIK** The central index key of a company, this is how the company relates to the EDGAR database.

- **Exchange** The exchange that the company is traded on (ex. New York Stock Exchange (NYSE) or NASDAQ).
- **Street1** The street address of the company headquarters.
- **Street2** If necessary and better description of the Street address of the company headquarters.
- **City** The city of the company headquarters.
- **State** The state of the company headquarters.
- **Zip** The zip code of the company headquarters.
- **State Description** Only applies to out of the Unites States companies.

The data pulled by the resource builder include both the financial data (Form 4 documents) and the market data. All these components are stored in the MySQL database, and the approach to do this is discussed in detail in the following two sections.

4.1.2 Market Data

Overview

The market data is gathered from the Yahoo financial database that is offered for free, this was a means to avoid one of the many other companies that charge large fees for access to their market data. This database has the historical data for all companies in the financial market. A note about the historical data is that the current day does not become "historical data" until the market closes (Market times in Pacific Standard Times are 6:30 a.m. to 1:00 p.m. [44]).

Yahoo Finance Database

The Yahoo database is accessed through an API service that Yahoo has built to get their market data. These endpoints are built to query for a specific company, the

date ranges, the format type, and the data asked for. The format type can be one of the many data transfer formats such as JSON, XML, or CSV. In this system the CSV is the file that the New Money Resource Builder accesses, and parses based off the comma separation. The data is not specified in this system since all the data is required for storage (these values are discussed later on). The date ranges are only used when the system needs to update the already stored data, since if a date is not provided all the data is returned. Then once this data is pulled it is handed off to the parser to store the data.

Historical Data

The Yahoo data is the Historical Data in the New Money Resource Builder. The historical data is the historical prices of a certain company throughout the time of the company being publicly traded. This is recorded information on each of the days the market is open. This information if useful to see daily moves (which can relate to information made public that day) or see yearly trends that develop over the companies lifetime (ex. company reports lower earnings in the summer time so prices always drop). The ability to see the trends in a companies history helps in the investment decisions, but having the data available in a database in a raw format gives the ability to run computational analysis on the data. The information provided by Yahoo for historical data is stored in the database. This data is a CSV form that is parsed by separation of the commas and formated to the double representation in the database. The historical_data table is the database representation of those values:

- **Day** The day associated the the date of the recorded historical prices.
- **Month** The month associated the the date of the recorded historical prices.
- **Year** The year associated the the date of the recorded historical prices.
- **Adjusted Close** The amendment to the closing price due to company transactions before the open of the next day [14].

- **Close** The price the stock closed at [15].
- **High** The highest price in the day the stock was at [15].
- **Low** The lowest price in the day the stock was at [15].
- **Open** The price the stock opened at [15].
- **Volume** The amount of shares of the company traded in the day [16].

4.1.3 Financial Data

Overview

The financial data comes from the EDGAR database, which is where the SEC files all the submitted forms received from the companies. This database will have all the financial information such as quarterly and yearly reports from companies, corporate announcements, and insider transactions. The latter is what will be focused on in this section since Form 4 is the highlight of the New Money system. The Form 4 requires parsing of the data that is in an XML format, and storage in the database requires the connections to be made to companies and insiders.

EDGAR

The EDGAR database is accessed through file transfer protocol (FTP) to grab the data in the file system. The SEC is in charge of the EDGAR system, and therefore files the submitted forms to the file system each day. A daily index is kept in EDGAR to show how many files are added to the system each day. This is useful for pulling information each day after the system is initialized. The initialization process of the system however pulls from an index file that is the combination of all forms submitted for each quarter of the year. This is referred to as the full index of EDGAR. To initialize the system with this it will parse through each year and each quarter of the year. Figure 4.1 shows the years that are stored in the system.

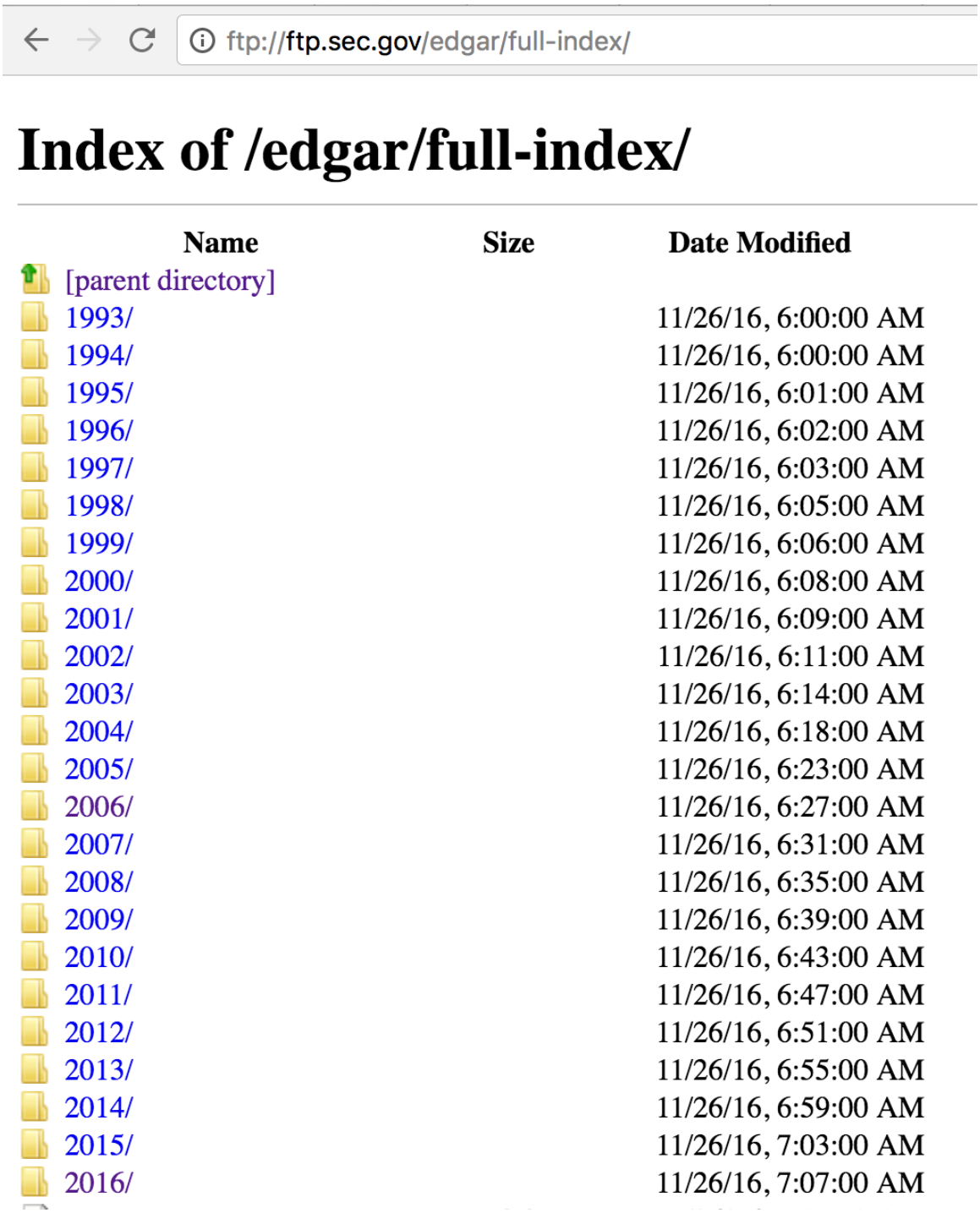
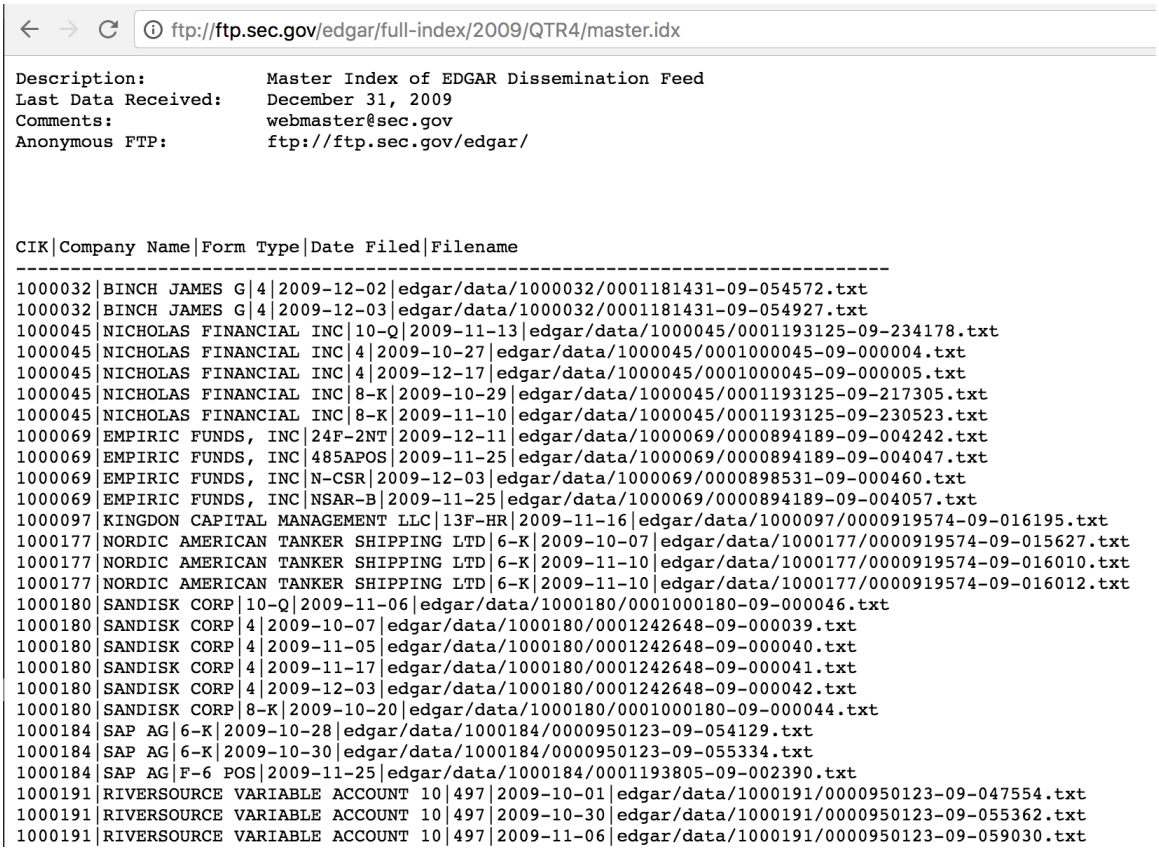


Figure 4.1: The full index page of the EDGAR database

In each quarter full index there is a file called the master index. This is the file that will give meta-data on all the filings in each quarter (Figure 4.2 has an example

of this). This single file is pulled and parsed by the system. Each line represents something that will be referred to as DailyData; this includes the company name, the CIK, the form type, and the file path to the form. Each piece of the line is stored in a DailyData object, and then placed in a Dictionary which has the key of the form type.



ftp://ftp.sec.gov/edgar/full-index/2009/QTR4/master.idx

```
Description:           Master Index of EDGAR Dissemination Feed
Last Data Received:    December 31, 2009
Comments:              webmaster@sec.gov
Anonymous FTP:         ftp://ftp.sec.gov/edgar/



CIK|Company Name|Form Type|Date Filed|Filename
--------------------------------------------------------------------------------
1000032|BINCH JAMES G|4|2009-12-02|edgar/data/1000032/0001181431-09-054572.txt
1000032|BINCH JAMES G|4|2009-12-03|edgar/data/1000032/0001181431-09-054927.txt
1000045|NICHOLAS FINANCIAL INC|10-Q|2009-11-13|edgar/data/1000045/0001193125-09-234178.txt
1000045|NICHOLAS FINANCIAL INC|4|2009-10-27|edgar/data/1000045/0001000045-09-000004.txt
1000045|NICHOLAS FINANCIAL INC|4|2009-12-17|edgar/data/1000045/0001000045-09-000005.txt
1000045|NICHOLAS FINANCIAL INC|8-K|2009-10-29|edgar/data/1000045/0001193125-09-217305.txt
1000045|NICHOLAS FINANCIAL INC|8-K|2009-11-10|edgar/data/1000045/0001193125-09-230523.txt
1000069|EMPIRIC FUNDS, INC|24F-2NT|2009-12-11|edgar/data/1000069/0000894189-09-004242.txt
1000069|EMPIRIC FUNDS, INC|485APOS|2009-11-25|edgar/data/1000069/0000894189-09-004047.txt
1000069|EMPIRIC FUNDS, INC|N-CSR|2009-12-03|edgar/data/1000069/0000898531-09-000460.txt
1000069|EMPIRIC FUNDS, INC|NSAR-B|2009-11-25|edgar/data/1000069/0000894189-09-004057.txt
1000097|KINGDON CAPITAL MANAGEMENT LLC|13F-HR|2009-11-16|edgar/data/1000097/0000919574-09-016195.txt
1000177|NORDIC AMERICAN TANKER SHIPPING LTD|6-K|2009-10-07|edgar/data/1000177/0000919574-09-015627.txt
1000177|NORDIC AMERICAN TANKER SHIPPING LTD|6-K|2009-11-10|edgar/data/1000177/0000919574-09-016010.txt
1000177|NORDIC AMERICAN TANKER SHIPPING LTD|6-K|2009-11-10|edgar/data/1000177/0000919574-09-016012.txt
1000180|SANDISK CORP|10-Q|2009-11-06|edgar/data/1000180/0001000180-09-000046.txt
1000180|SANDISK CORP|4|2009-10-07|edgar/data/1000180/0001242648-09-000039.txt
1000180|SANDISK CORP|4|2009-11-05|edgar/data/1000180/0001242648-09-000040.txt
1000180|SANDISK CORP|4|2009-11-17|edgar/data/1000180/0001242648-09-000041.txt
1000180|SANDISK CORP|4|2009-12-03|edgar/data/1000180/0001242648-09-000042.txt
1000180|SANDISK CORP|8-K|2009-10-20|edgar/data/1000180/0001000180-09-000044.txt
1000184|SAP AG|6-K|2009-10-28|edgar/data/1000184/0000950123-09-054129.txt
1000184|SAP AG|6-K|2009-10-30|edgar/data/1000184/0000950123-09-055334.txt
1000184|SAP AG|F-6 POS|2009-11-25|edgar/data/1000184/0001193805-09-002390.txt
1000191|RIVERSOURCE VARIABLE ACCOUNT 10|497|2009-10-01|edgar/data/1000191/0000950123-09-047554.txt
1000191|RIVERSOURCE VARIABLE ACCOUNT 10|497|2009-10-30|edgar/data/1000191/0000950123-09-055362.txt
1000191|RIVERSOURCE VARIABLE ACCOUNT 10|497|2009-11-06|edgar/data/1000191/0000950123-09-059030.txt
```

Figure 4.2: The Master index of a single quarter filed in the EDGAR database

Once the EDGAR master index is fully parsed, the processing begins. Java will process the information on an Interface class for form parsing. The interface class requires a List of the DailyData to be assigned to it in “Init”. This list is consistent for all the form types so the interface can variable can be assigned to any form type parser. The class becomes reassigned after the category of the form type is determined. This is done with having Java create a new instance of the class based off a string that is created from the form name. Each form type will have a class that implements the

general form parser class. The "Init" of the specific form class is called to run the processing of the collected master index data.

Form 4

The Form 4 is the only document in the EDGAR database that is parsed and stored in the New Money Resource Builder. This form was picked because it is the most filed in the EDGAR database, and having this information stored is beneficial to the traders. The file pulled from the EDGAR database for a Form 4 is in eXtensible Markup Language (XML). This gives a standard to each file and allows parsing of the document possible without any errors or edge cases. The XML form is setup by the SEC and follows a standard that can be viewed in the guide sheet [27]. Figure 4.3 shows an example of the XML; from this it can be seen the company being traded is Apple Inc. by the Senior Vice President Donald Rosenberg. The trade type is on a derivative of restricted stock.

Figure 4.3: Part of the XML format of the Form 4 document that is pulled from EDGAR

The data from the XML is parsed using a DOM tree provided by Java [12]. The DOM tree allows access to nested values and iterations on these values. Recursion on the elements of the XML document will grab all the required nested values. Once the

recursive iterations find a function in the class by the same name as the name, that function is called. Once in the function it is possible to iterate through all the child elements and grab the values from them to be stored in the object. These levels that are called include the insiders that are trading on the form, and the trades executed on the form.

The database at this level has four piece to save a From 4 in New Money Data Vending. An insider table is built to create an insider; information on the insider is all the same as a company minus the exchange and ticker. An insider header that will relate an insider to the filed transactions; this header points to an insider, then save the relation the insider has to the company, and the date this was filed. There are two types securities that can be traded on, derivative and non-derivative. In those two there will either be an acquire or disposal of the security. The two types of trades have nearly identical tables, but derivative transactions contain a conversion or exercise price, transaction price per share, exercise date, expiration date, underlying security title, underlying security shares, and underlying security value. These two types also have a potential to have a footnote attached to each value, so the footnote must also be saved in the database. Below is a list of all the values that a transaction can have [29].

- **Security Title** The name of the security type (ex. Common Stock A).
- **Conversion or Exercise Price** The price to the exchange from one convertible asset to the another (Derivative Only).
- **Transaction Date** The date the transaction took place.
- **Deemed Execution Date** The date the execution takes place only if calculation is pursuant.
- **Transaction Form Type** The type of form being file (in all cases for this it is 4).

- **Transaction Code** The type of transaction that is being done (ex. gifted, open market, etc.).
- **Equity Swap Involved** This stated if the transaction is a future, so transaction will happen later.
- **Transaction Timeliness** If the transaction was either early or late.
- **Transaction Shares** The shares involved in the transaction.
- **Transaction Total Value** The total value of of the transaction.
- **Transaction Price Per Share** The price of each share involved (Derivative Only).
- **Transaction Acquired Disposed Code** If the transaction was a purchase or sale.
- **Exercise Date** The date the derivative is conducted (Derivative Only).
- **Expiration Date** The date the derivative is called upon (Derivative Only).
- **Underlying Security Title** The name of the underlying security being traded (Derivative Only).
- **Underlying Security Shares** The shares of the underlying security being traded (Derivative Only).
- **Underlying Security Value** The value of the shares of the underlying security being traded (Derivative Only).
- **Shares Owned Following Transaction** The shares owned of the asset after the transaction.
- **Value Owned Following Transaction** The value owned of the asset after the transaction.

- **Direct or Indirect Ownership** If the shares are held personally or elsewhere.
- **Nature Of Ownership** Who conducted the transaction.
- **Footnotes** The footnotes are associated to all the above, and there is a footnote for each one saved in the database.

4.2 New Money RESTful Service

4.2.1 Overview

The New Money RESTful Service is in charge of sending the data from the database. This information is the market and financial data that was built in the resource builder. The rest service is built using Java Spring Boot for the simplicity of setup, the predefined networking capabilities, and the database interface. The rest service has various endpoints to hit that send the data back to the requester. These endpoints come in the form of controllers. Figure 4.4 is one endpoint that will return a JSON list. This particular function is building a list of insiders that belong to a specified company. The models are the values that are returned as the JSON values, and these models are the same structure as the database which was discussed in Section 4.1 and is in Figure 3.10. A note to add to the rest section is this service only provides GET requests, no endpoints have been created to POST data to the database. This section will go over the endpoints that are integrated to the New Money RESTful Service.

```
75    @GetMapping(value = "/company/{ticker}/insiders")
76    public @ResponseBody Set<Insider> getCompanyInsiders(@PathVariable("ticker") String stockId) {
77      Set<Insider> insiders = new HashSet<Insider>();
78      for(InsiderForm4 insiderForm : companyRepository.findByTicker(stockId).getInsiderForm4()) {
79        insiders.add(insiderForm.getInsider());
80      }
81      return insiders;
82    }
```

Figure 4.4: This is an example REST controller endpoint that will build a list of all the insiders for a certain company

4.2.2 Requests

The requests on the RESTful service is aided by Java Spring Boot. The Spring Boot framework creates the database connection with objects called repositories. These extend a create-read-update-delete (CRUD) class that will get requested information from the associated database. The endpoint are also handled by the Spring Boot, so each URL mapping and connected values will be handled with the Spring Boot.

Get All Companies

This controller endpoint will return a list of all the companies in the database. This list is a JSON format with all the meta-data associated to each company.

Get Company

This controller endpoint will return a JSON that is the single company requested. The JSON will have all the meta-data for the one particular company. The URL requires a path variable to look up the company in the database; that path variable is the ticker of the company.

Get Insiders and Relations on a Company

This controller builds a JSON of the insiders on a company and withing the insiders is details of all the transaction that are associated to them. A note about that transactions is that it is not the particular data on the transaction itself, but the fact that there was a transaction, the date that the transaction was filed, and the position the insider has to the company. The URL for this requires a path variable which is the ticker of the company.

Get Non-derivative Forms for a Specific Transaction

This controller builds a list of all the non-derivative that relate to a specific transaction form. This is a single Form 4 may have multiple non-derivative transactions, so this is all those transactions in a JSON list of all the data related to the non-derivative trade. This URL requires the id built for the certain Form 4 document.

Get Derivative Forms for a Specific Transaction

Same as the non-derivative controller. This controller builds a list of all the derivative that relate to a specific transaction form. This is a single Form 4 may have multiple derivative transactions, so this is all those transactions in a JSON list of all the data related to the derivative trade. This URL requires the id built for the certain Form 4 document.

Get Historical Data for a Specific Company

This controller builds a JSON list of each historical data meta-data for a specific company. The URL requires the company ticker.

Get Insider

This controller endpoint will return a specific insider and the meta-data for that insider. This also includes the transactions that were made (again a note on this; transactions are only dates and positions). The URL for this requires the insider CIK.

Get All Insiders

This controller endpoint will return a list of all insiders that are stored in the database. This is a JSON format with all the meta-data associated to each insider.

Get All Trades on an Insider

This controller endpoint will return the transactions that were made by a single insider. This URL requires the CIK of the insider.

Get Company of a Transaction

This controller endpoint will get the transactions company. This is the company meta-data that is being traded in a certain transaction. The URL for this requires the header value for the transactions.

4.3 New Money Website

4.3.1 Overview

The New Money Website is built in Java Spring Boot to take advantage of the web features provided. This includes the controller endpoints to map to a path, the models that are returned to the views, the template views that can be built with thymeleaf, and the functions to call rest API services. The website follows a model-view-controller architecture; the models are the models are the data stored in the database that was built using the New Money Resource Builder, the controllers will read in the requests and return the view, and the views are the front-end provided to the user.

The controller is the endpoints that are hit by the front-end user, this controller will call the New Money Rest Service, set the models to the template view, and render the view to the front-end web browser. Figure 4.5 is one such endpoint that will build a view page. The particular function hits the rest service for the specified company and the historical data of that company. This function will then add those models as attributes to the view, which renders the page using thymeleaf. When required to return a model to the view, the controller is the interaction to the New Money Rest Service, and builds the models.

```
41    @GetMapping(value = "/company/chart/{ticker}")
42    public String getCompanyChart(@PathVariable("ticker") String stockId, Model model) {
43      RestTemplate restTemplate = new RestTemplate();
44          Company company = restTemplate.getForObject("http://localhost:8080/company/" + stockId, Company.class);
45          HistoricalDatum [] historicalData = restTemplate.getForObject("http://localhost:8080/historicaldata/"
46                                                                        + stockId, HistoricalDatum[].class);
47
48          model.addAttribute("historicalData", Arrays.asList(historicalData));
49      model.addAttribute("company", company);
50      return "company/chart";
51    }
```

Figure 4.5: controller example

The view portion of the New Money Website is built with HTML, CSS, and Javascript. To make styling easier with some pre-built views Bootstrap was used. With Spring Boot rendering the views has a method to build the templates. Thymeleaf is the library on the Java Spring Boot that helps in rendering the template pages.

Thymeleaf takes the models and will build them into the HTML and Javascript pages when accessed. This happens on the server side, so the page is requested by the front-end and the returned HTML document is built server side and returned with the correct values in the HTML document. Figure 4.6 is a code snippet of thymeleaf in HTML. The thyme leaf is looping across an array, and building a table of the historical data with formatting on the integers and double variables.

```
<div class="card">
  <div id="raw-data" class="collapse" role="tabpanel" aria-labelledby="RawData">
    <div class="card-block">
        <table id="sorted-table" class="table table-striped table-bordered">
          <thead>
            <tr>
              <th>Date</th>
              <th>Open</th>
              <th>High</th>
              <th>Low</th>
              <th>Close</th>
              <th>Volume</th>
              <th>Adjusted Close</th>
            </tr>
          </thead>
          <tr th:each="hd : ${historicalData}">
            <td th:inline="text">[[${hd.year}]]/[[${hd.month}]]/[[${hd.day}]]</td>
            <td th:text="${#numbers.formatDecimal(hd.open,1,2)}">2</td>
            <td th:text="${#numbers.formatDecimal(hd.high,1,2)}">3</td>
            <td th:text="${#numbers.formatDecimal(hd.low,1,2)}">4</td>
            <td th:text="${#numbers.formatDecimal(hd.close,1,2)}">5</td>
            <td th:text="${#numbers.formatInteger(hd.volume,3,'COMMA')}">6</td>
            <td th:text="${#numbers.formatDecimal(hd.adjustedClose,1,2)}">7</td>
          </tr>
        </table>
    </div>
  </div>
</div>
```

Figure 4.6: An example of thymeleaf in an HTML document; building a table off of a list of historical data

The core components to the view are the navigations. There is a top navigation to get to the selection of companies or insiders. The side navigation bar is for navigating once insider a company. The side bar will offer a path back to the main company page, or the insiders on the company. Figure 4.7 is the home page, which shows off the top and side bars.

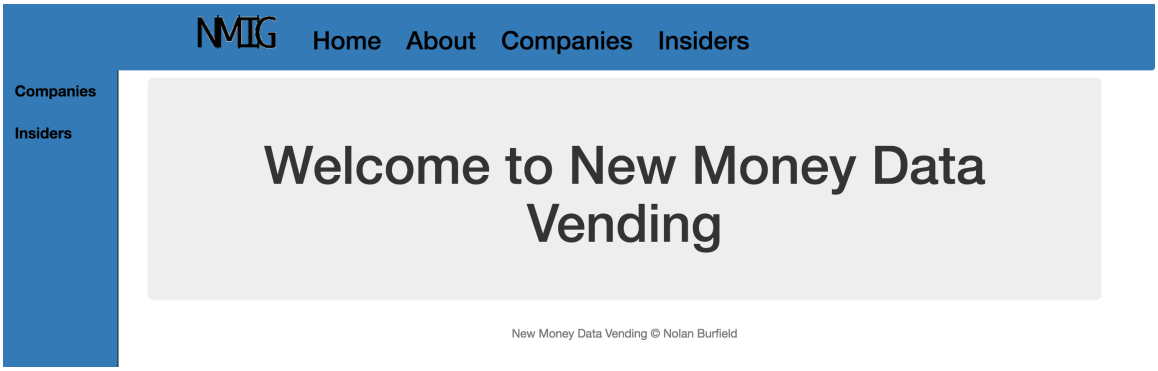


Figure 4.7: The New Money Data Vending home page

The following few sections will go over the views that are provided by the New Money Website to showcase some of the data that is stored in the database.

4.3.2 Company

The company portion of the website has three distinct points to a company view. The user will first navigate to the company page, which will then provide a list of companies that are stored in the system (Figure 4.8 shows the table). The user then can select a company to view the stored data on that company. The company page will provide links in the left navigation bar to select the company home page, or the insiders on the company. The main content of the page has company details, historical data chats, and historical data raw table. The company has meta-data associated to it, so that is presented on the top of the page (Figure 4.9) in what is called the company card. The user can toggle on and off the chart and data tables.

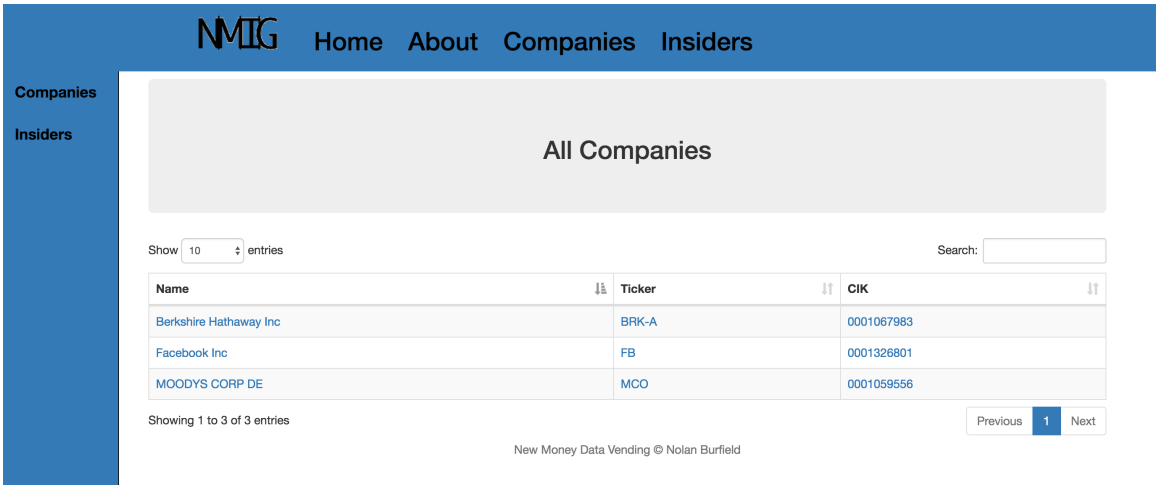


Figure 4.8: A view of all the companies in a table

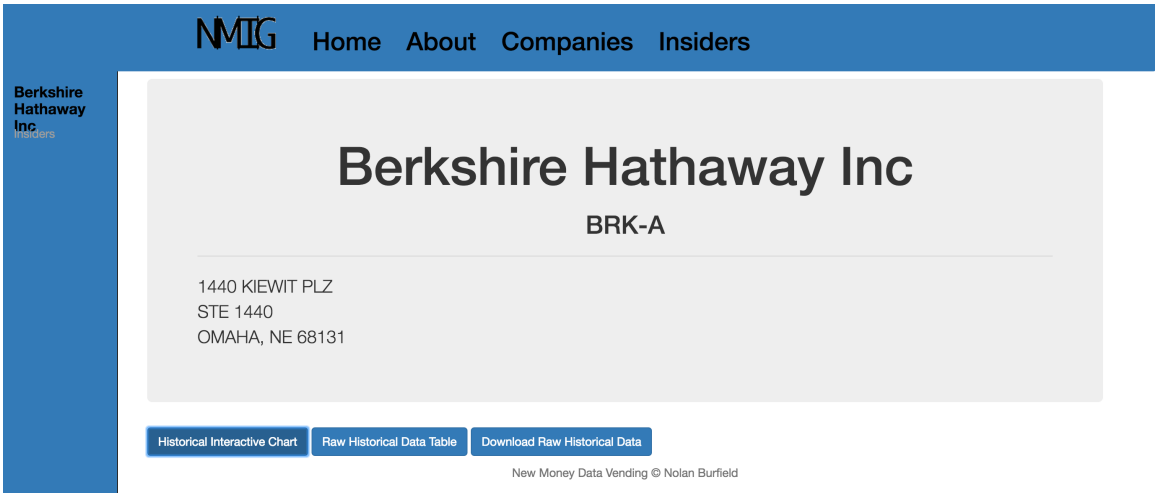


Figure 4.9: The top view of a page showing Berkshire Hathaway company meta-data

Historical Data

The historical data for a company is displayed in both a graph and raw data table. The graph and table both can be toggled on and off, and the data is also able to be downloaded in a CSV file. The controllers for the display of this will get both the company data, and the historical data and return these to the view. The controller endpoint to the CSV download will return to the user a file format which inherently is downloaded by the users browser. Figure 4.10 is a top portion of the downloaded data.

```
1   date,open,high,low,close,volume,adjusted_close
2   2009-7-22,27.0,27.379999,26.33,26.52,6836600.0,23.666479
3   2004-2-9,64.699997,65.489998,64.139999,64.199997,1321800.0,27.570901
4   2001-8-14,33.700001,33.990002,33.630001,33.869999,373000.0,14.397261
5   1998-8-26,19.375,19.375,19.0,19.125,1827200.0,8.090633
6   2003-8-18,52.150002,52.5,52.040001,52.299999,477800.0,22.442731
7   1998-2-23,24.75,25.0,24.5,24.625,760400.0,10.417351
8   1998-4-13,26.0,26.0,25.375,25.75,1286200.0,10.893271
9   1995-9-29,16.25,16.375,16.25,16.375,2118000.0,6.927274
10  2012-8-30,38.540001,39.25,38.52,39.060001,1236400.0,36.803102
11  1999-7-6,27.25,27.25,26.125,26.125,635200.0,11.051911
12  2005-2-28,85.099998,85.099998,83.730003,83.910004,1313600.0,36.192789
13  2015-7-30,110.209999,111.760002,109.730003,111.529999,643500.0,109.089313
```

Figure 4.10: A CSV of the historical data downloaded from New Money Website

The graph is rendered on the same page as the company information, but in an iframe. This is to allow the Javascript associated to the TechanJS (Javascript library to build charts) to render full screen. This iframe has an independent endpoint on the controller that will grab the company data and the historical data and send the model data to the view for the graph. The TechanJS graph is in a candlestick format which shows off the historical data in an efficient way. A description of what a candlestick graph is can be found below. The graph in Figure 4.11 is a full graph of Facebook Inc. The data in the full view is not that readable, but the graph offers a zoom option, so the use can zoom into certain points in the companies time and see price-points better in that view (Figure 4.12).

Figure 4.11: The full data of all historical data points on Facebook Inc.

Figure 4.12: The Facebook Inc. graph zoomed in to limit the time period shown and better see the candlestick

The candlestick is a common representation of stock data in a chart. This format show the open, high, low, and close of a stock in a single day with one point on a graph, and example can be seen in Figure 4.13. The days representation is a box with two line on the top and bottom. The line out of the top will stop at the "high" of

the day. The line out of the bottom will stop at the “low” of the day. The box will represent the “open” and “close” of the stock, with the top and bottom of the box representing those values. Depending on the color of the box will represent a drop or gain in the stock price for the day. If a drop in the price occurred then the open will be the top of the box, and the bottom of the box will be the close. In the case of the candlestick graph in this paper red is a drop, and green is a gain [37].

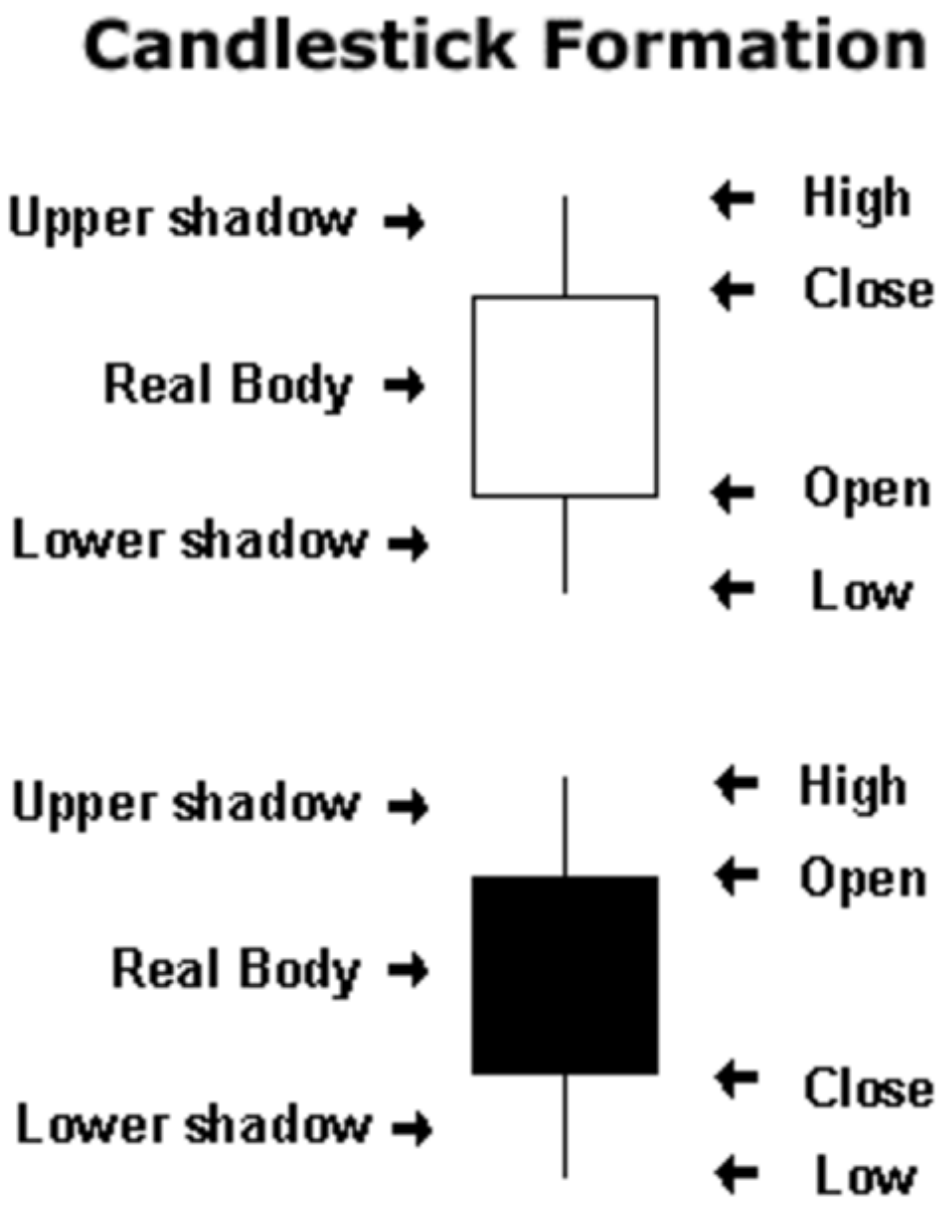
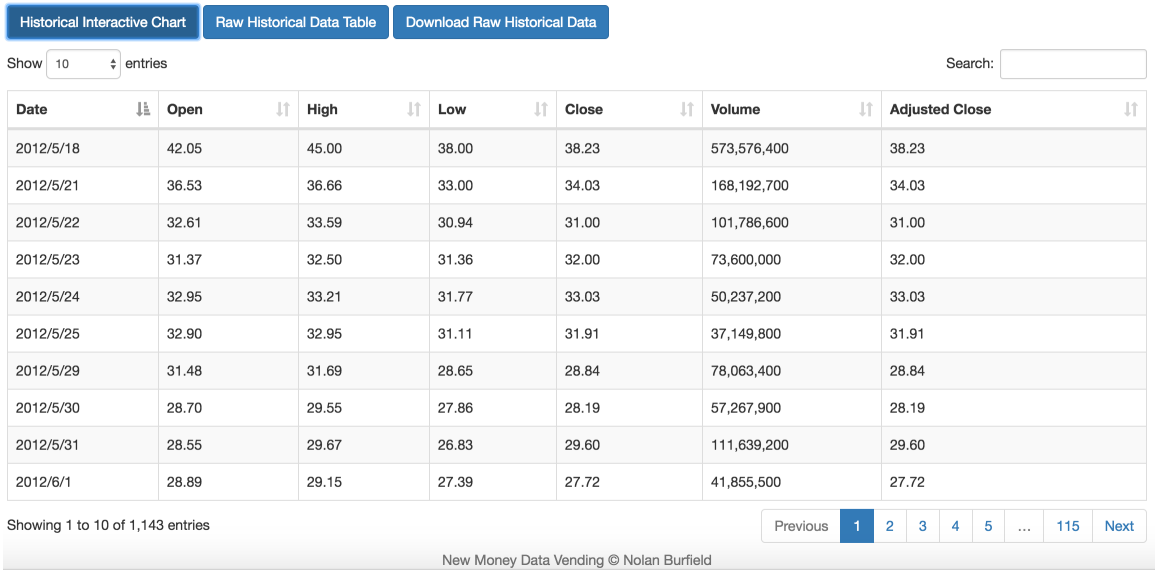


Figure 4.13: The two types of candlestick values for a single day, showing gain or loss [37]

The raw data for the historical information on a company is displayed in a table on the HTML page. The table is built with Bootstrap features, so the table is searchable (helps to find dates), sortable by value of all columns, and the size of the table can be limited or expanded to show the raw data rows. Providing the user with a

table will allow the comparison of the figure and raw data side by side to have the ability to look up exact values on certain dates. The Figure 4.14 shows the table in the page.



Date	Open	High	Low	Close	Volume	Adjusted Close
2012/5/18	42.05	45.00	38.00	38.23	573,576,400	38.23
2012/5/21	36.53	36.66	33.00	34.03	168,192,700	34.03
2012/5/22	32.61	33.59	30.94	31.00	101,786,600	31.00
2012/5/23	31.37	32.50	31.36	32.00	73,600,000	32.00
2012/5/24	32.95	33.21	31.77	33.03	50,237,200	33.03
2012/5/25	32.90	32.95	31.11	31.91	37,149,800	31.91
2012/5/29	31.48	31.69	28.65	28.84	78,063,400	28.84
2012/5/30	28.70	29.55	27.86	28.19	57,267,900	28.19
2012/5/31	28.55	29.67	26.83	29.60	111,639,200	29.60
2012/6/1	28.89	29.15	27.39	27.72	41,855,500	27.72

Figure 4.14: A table built of the raw data, the table is limited to only showing 10 values for readability

Financial Data

The financial data controller calls the rest endpoint to grab all insiders on a company. This will return the JSON with the users and their transactions in relation to the company. It is necessary to make a call to the rest service to grab the derivative and non-derivative forms. The view will build the company portfolio again and a table showing the transactions. The table has the insider name, and below them is the relation to the company, and then the trade that was performed. Due to the nature of the trades there is not much consistency for the Form 4 tables that are filled or not, so there is limited information provided to the user in the table This includes the name of the stock traded, the date it was traded, and if it was purchased or sold. Figure 4.15 shows the table for a company.

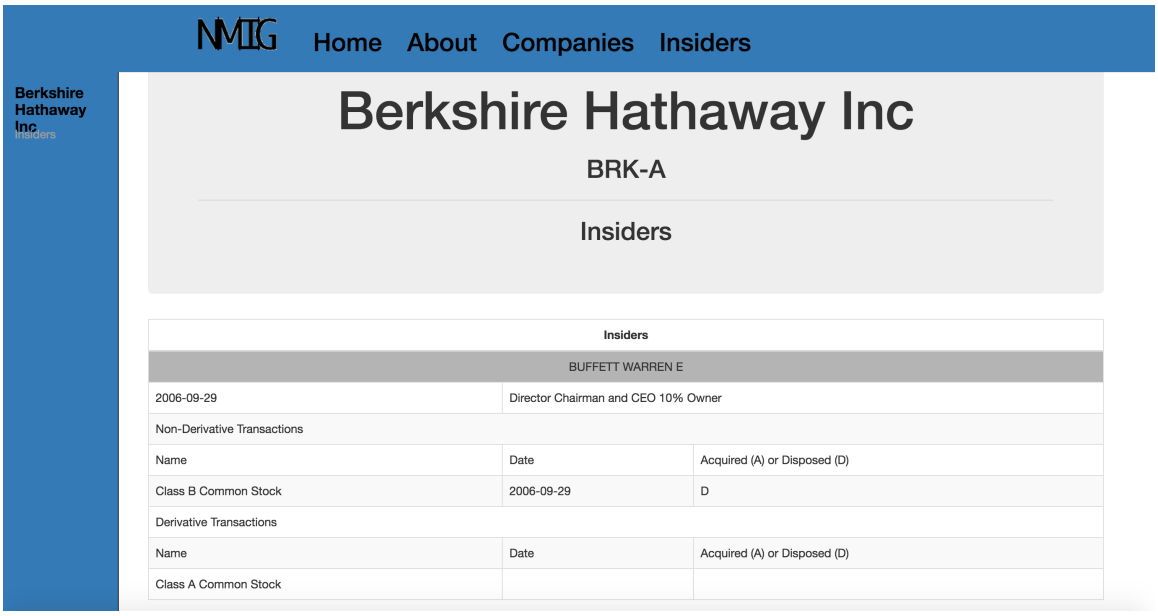


Figure 4.15: A table of the insider transactions on a company; this is showing multiple owners and their trade made

4.3.3 Insiders

Insiders are collected and stored from the Form 4 documents. A lot of times insiders apply to multiple trades among multiple companies. New Money Website allows the selection of a single insider to view all their transactions. Figure 4.16 shows the ability to select an insider that is stored in the database. Once an insider is selected their profile page is provided to the front-end. This requires the controller to grab the meta-data on an insider, and all their transactions. The transactions must be associated to a company, so the controller grabs the company meta-data. Then each transaction will need to be associated to the derivative and non-derivative trades made inside each company.

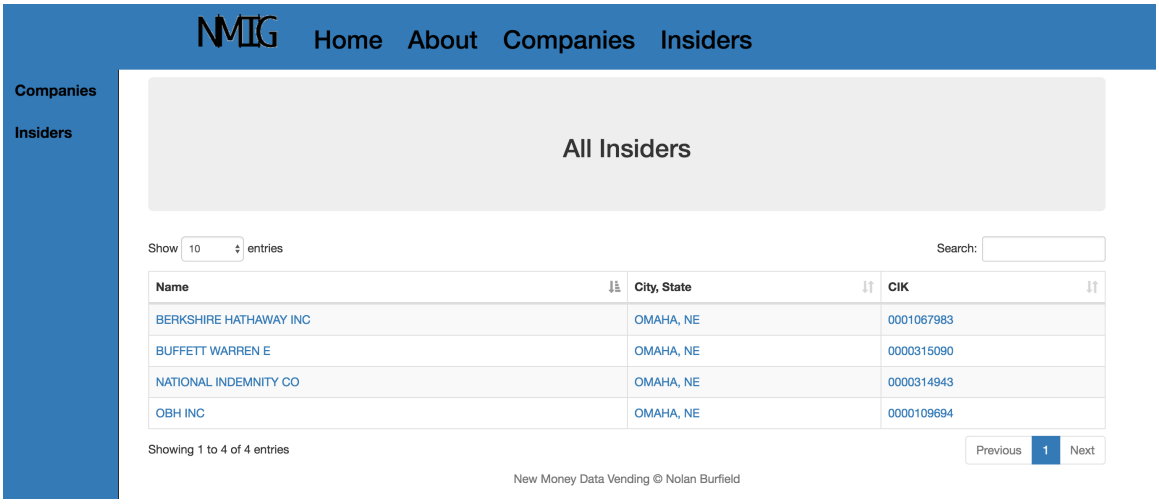


Figure 4.16: A view of all the insiders in a table

The insider view will provide the same profile page as a company does, showing off the insider stored meta-data. Then the list of transactions to the insider is presented in a table. This table is organized by the companies that trades took place in. Figure 4.17 shows this page describe for Warren Buffett and the transactions on two companies, Berkshire Hathaway and Moodys Corp.

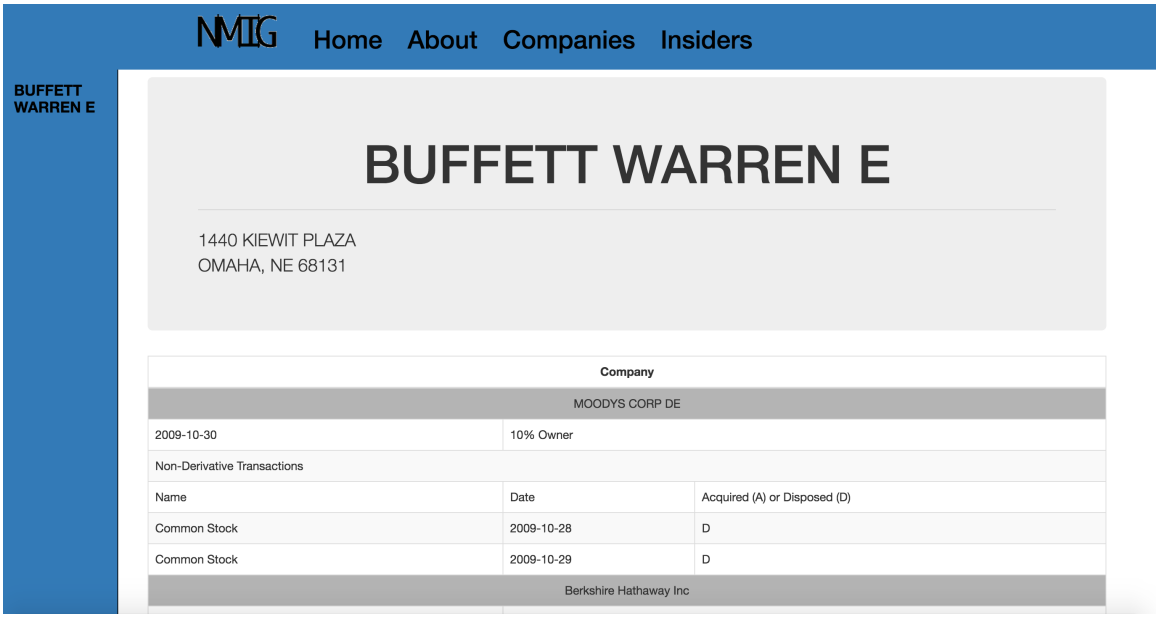


Figure 4.17: A view of a single insider selected showing their transactions among companies

Chapter 5

Conclusions and Future Work

5.1 Conclusion

The system described in this thesis, New Money Data Vending, focuses on the start of a system to provide both market and financial data to users. The idea of the system is to automate all the data collection and form parsing. The collected data is stored in a database, and provided to users through a RESTful service or a front-end website. The whole system back-end was developed with Java Spring, and the front-end is a web application. The focus of the data provided to the user was historical data and the Form 4 documents that the Securities and Exchange Commission requires all publicly traded companies to file.

The system is relevant and useful due to the growing capabilities of computational systems. While the stock market is not a new concept, information access speed through computation is relatively new. Computer speeds are growing, and the ability of data vending was an old market dominated by large corporations charging large fees. The automation of data collection, storage, and vending is useful to all large financial corporations (ex. mutual funds or hedge funds).

5.2 Future Work

The New Money Data Vending system is in the very beginning stages of development, and requires many more features and enhancements to provide the best data vending system for users.

5.2.1 Resource Builder

The resource builder will be expanded to include the parsing of more forms from the EDGAR database. The next form to focus on the the earnings reported each quarter by companies. These earnings values will allow for the historical data to interact with the financial data and allow for the calculations of price per share and revenue streams.

5.2.2 Restful Service

With more features being implemented in the back-end there is a need to display this in the front-end. The rest service will need to add more endpoints to allow for the return of the new data in the database. Not only will the rest service provide JSON forms of the database data, but it will need to consume requests to run analytics of the database data. These analytics will be built by the users, and queued on a separate server. This service will be the analytics portion on New Money Data Vending.

5.2.3 Website

As with the rest service needing updates when the database updates, the front end will need to create new endpoints to view the data. With that the website will need to be updated to have user login, and user sessions. The front-end will also be cleaned up to compete with the modern websites with dynamic page loading and rendering.

Bibliography

[1] Apache. Apache commons net. [Accessed on 26 November 2016]. URL: https://commons.apache.org/proper/commons-net/.

[2] Apache. Welcome to apache maven. [Accessed on 26 November 2016]. URL: https://maven.apache.org.

[3] Jan Bodnar. Mysql java tutorial. [Accessed on 26 November 2016]. URL: http://zetcode.com/db/mysqljava/.

[4] Bootstrap. About bootstrap. [Accessed on 26 November 2016]. URL: http://getbootstrap.com/about/.

[5] Ernie Chan. *Quantitative trading: how to build your own algorithmic trading business*. Volume 430. John Wiley & Sons, 2009.

[6] Andre Dumas. Techanjs. [Accessed on 26 November 2016]. URL: http://techanjs.org/.

[7] Eclipse. About the eclipse foundation. [Accessed on 26 November 2016]. URL: https://eclipse.org/ide/.

[8] Kelly Elias. Download stock ticker symbols. [Accessed on 26 November 2016]. URL: http://www.jarloo.com/download-stock-ticker-symbols/.

[9] Federal Deposit Insurance Corporation. Fdic: federal deposit insurance corporation. [Accessed on 26 November 2016]. URL: https://www.fdic.gov/.

[10] Gallup. In u.s., 54% have stock market investments, lowest since 1999. [Accessed on 26 November 2016]. URL: http://www.gallup.com/poll/147206/stock-market-investments-lowest-1999.aspx.

[11] Diego García and Øyvind Norli. Crawling edgar. *The spanish review of financial economics*, 10(1):1–10, 2012.

[12] Lokesh Gupta. Java xml dom parser example tutorial. [Accessed on 26 November 2016]. URL: http://howtodoinjava.com/xml/java-xml-dom-parser-example-tutorial/.

[13] Yves Hilpisch. *Python for finance: analyze big financial data.* " O'Reilly Media, Inc.", 2014.

[14] Investopedia. Adjusted closing price. [Accessed on 26 November 2016]. URL: http://www.investopedia.com/terms/a/adjusted_closing_price.asp?lgl=no-infinite.

[15] Investopedia. Ohlc chart. [Accessed on 26 November 2016]. URL: http://www.investopedia.com/terms/o/ohlcchart.asp?lgl=no-infinite.

[16] Investopedia. Volume. [Accessed on 26 November 2016]. URL: http://www.investopedia.com/terms/v/volume.asp?lgl=no-infinite.

[17] Shivprasad Koirala and Marla Sukesh. Learn mvc (model view controller) step by step in 7 days day 1. [Accessed on 26 November 2016]. URL: http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in.

[18] Douglas A Lyon. Multi-threaded data mining of edgar ciks (central index keys) from ticker symbols. In *Parallel and distributed processing, 2008. ipdps 2008. ieee international symposium on*. IEEE, 2008, pages 1–7.

[19] NASDAQ. Nasdaq data-on-demand. [Accessed on 26 November 2016]. URL: http://www.nasdaqdod.com/.

[20] Oracle. About jdbc resources and connection pools. (sun java system application server platform edition 8.2 administration guide). [Accessed on 26 November 2016]. URL: https://docs.oracle.com/cd/E19830-01/819-4712/ablii/index.html.

[21] Oracle. Java platform se 8. [Accessed on 26 November 2016]. URL: http://docs.oracle.com/javase/8/docs/api/.

[22] Oracle. Mysql 8.0 reference manual. [Accessed on 26 November 2016]. URL: http://dev.mysql.com/doc/refman/8.0/en/introduction.html.

[23] Oracle. The java tutorials. [Accessed on 26 November 2016]. URL: https://docs.oracle.com/javase/tutorial/.

[24] Pivotal Software. Introduction to the spring framework. [Accessed on 26 November 2016]. URL: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html.

[25] Scottrade. Trading fees, investment fees, and other transaction prices. [Accessed on 26 November 2016]. URL: https://www.scottrade.com/online-brokerage/trading-fees-commissions.html?icid=8|180|1010|71.

[26] Securities and Exchange Commission. Analyzing analyst recommendations. [Accessed on 26 November 2016]. URL: https://www.sec.gov/investor/pubs/analysts.htm.

[27] Securities and Exchange Commission. Edgar ownership xml technical specification (version 5.1). [Accessed on 26 November 2016]. URL: https://www.sec.gov/info/edgar/ownershipxmltechspec.htm.

[28] Securities and Exchange Commission. Fast answers. [Accessed on 26 November 2016]. URL: https://www.sec.gov/answers/form345.htm.

[29] Securities and Exchange Commission. Form 4. [Accessed on 26 November 2016]. URL: https://www.sec.gov/about/forms/form4data.pdf.

[30] Securities and Exchange Commission. Information for ftp users. [Accessed on 26 November 2016]. URL: https://www.sec.gov/edgar/searchedgar/ftpusers.htm.

[31] Securities and Exchange Commission. Using edgar - researching public companies. [Accessed on 26 November 2016]. URL: https://www.investor.gov/researching-managing-investments/researching-investments/using-edgar-researching-public-companies.

[32] Securities and Exchange Commission. What we do. [Accessed on 26 November 2016]. URL: https://www.sec.gov/about/whatwedo.shtml.

[33] Zachary M. Seward. This is how much a bloomberg terminal costs. [Accessed on 26 November 2016]. URL: http://qz.com/84961/this-is-how-much-a-bloomberg-terminal-costs/.

[34] Kathy Sierra and Bert Bates. *Head first java.* " O'Reilly Media, Inc.", 2005.

[35] Raja H. Singh, Nolan Burfield, and Frederick Harris Jr. Data retrieval and parsing of form 4 from the edgar system using multiple cpus, 2016.

[36] Raja H. Singh, Nolan Burfield, and Frederick Harris Jr. Market data extractor (mdx): a system to download market data, 2015.

[37] StockCharts. Introduction to candlesticks. [Accessed on 26 November 2016]. URL: http://stockcharts.com/school/doku.php?id=chart_school:chart_analysis:introduction_to_candlesticks.

[38] The Thymeleaf Team. Thymeleaf. [Accessed on 26 November 2016]. URL: http://www.thymeleaf.org/.

[39] Meltem Sönmez Turan, Elaine B Barker, William E Burr, and Lidong Chen. Sp 800-132. recommendation for password-based key derivation: part 1: storage applications, 2010.

[40] Tutorialspoint. Jdbc tutorial. [Accessed on 26 November 2016]. URL: http://www.tutorialspoint.com/jdbc/.

[41] W3Schools. Css introduction. [Accessed on 26 November 2016]. URL: http://www.w3schools.com/css/css_intro.asp.

[42] W3Schools. Html introduction. [Accessed on 26 November 2016]. URL: http://www.w3schools.com/html/html_intro.asp.

[43] W3Schools. Javascript introduction. [Accessed on 26 November 2016]. URL: http://www.w3schools.com/js/js_intro.asp.

[44] Wall Street Daily. Nyse holiday closings 2016-2017. [Accessed on 26 November 2016]. URL: http://www.wallstreetdaily.com/nyse-stock-market-holiday-closings-schedule/.

[45] Phillip Webb, Dave Syer, Josh Long, Stephane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, and Sebastien Deleuze. Spring boot reference guide. [Accessed on 26 November 2016]. URL: http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#boot-documentation-about.

[46] Worldbank. Market capitalization of listed companies (current us$). [Accessed on 26 November 2016]. URL: http://data.worldbank.org/indicator/CM.MKT.LCAP.CD.

[47] XIGNITE. Market data feed and api. [Accessed on 26 November 2016]. URL: http://www.xignite.com/.

[48] Xignite. Support. [Accessed on 26 November 2016]. URL: http://www.xignite.com/Support/FAQ.aspx?faqtype=Topics&faqcat=Subscriptions#270.

[49] Yahoo. Yahoo query language (yql). [Accessed on 26 November 2016]. URL: https://developer.yahoo.com/yql/.