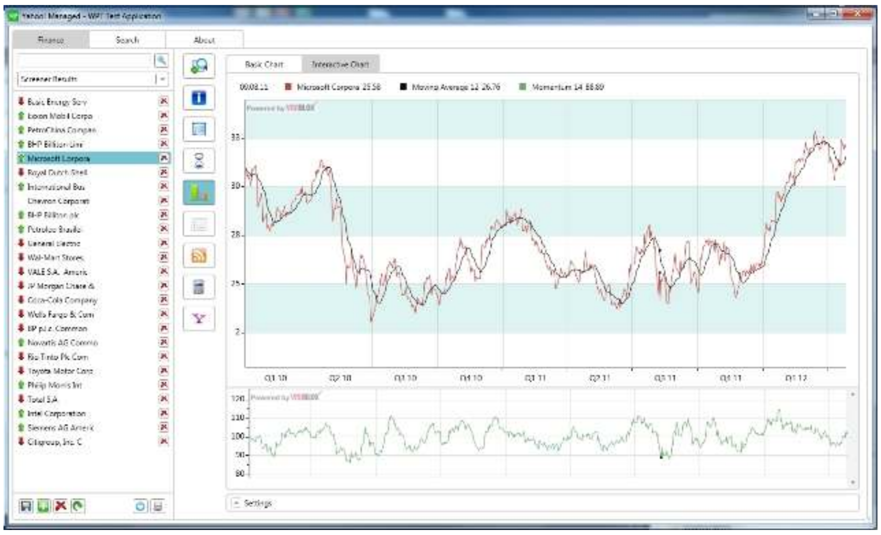
Yahoo! Managed

MaasOne, 1 May 2012 Rate:

★★★★★ 4.87 (45 votes)

Download financial data, managing online portfolio or using Search BOSS from Yahoo! with .NET

- **Download Release**
- **Download Source code**
- **Download Showcase**
- **Download Documentation**
- **Project Homepage**



Basic Energy Serv
Exxon Mobil Corpo
PetroChina Compan
BHP Billiton Lim
Microsoft Corpora
Royal Dutch Shell
International Bus
Chevron Corporat
BHP Billiton plc
Petroleo Brasile
General Electric
Wal-Mart Stores
VALE S.A. Americ
JP Morgan Chase &
Coca-Cola Company
Wells Fargo & Com
BP p.l.c. Common
Novartis AG Commo
Rio Tinto Plc Com
Toyota Motor Corp
Philip Morris Int
Total S.A.
Intel Corporation
Siemens AG Americ
Citigroup, Inc. C

Introduction

The web service that Yahoo! provides is nice, but you have to know the right tags and symbols and how to build the URL to use it. This library will undertake that annoying task for you and returns the received data in managed classes.

This article provides the basic understanding and appliance of the library with the aid of some examples. For actual informations, releases or documentation etc. look at the project homepage.

Using the Code

Every example in this article needs the following imports:

⊟ Collapse | Copy Code

```
using MaasOne;
using MaasOne.Base;
```

Download Classes and Namespaces

Every example in this section needs the following imports:

⊟ Collapse | Copy Code

```
using MaasOne.Finance.YahooFinance;
```

The library provides a lot of different classes for downloading data, but with every download class it's the same principle. Each class inherits from the generic `Base.DownloadClient<T>` class. This class provides the basic functions for controlling the download session and getting the data. The only differences of each download class are the `Settings` property and the result class that is of type `T`. Every download class has a `Settings` property with a specialized settings class. This class inherits everytime from `Base.SettingsBase`. The result object can be of every type and stores the managed result data. There is a standardization of names in this library. E.g., if the download class is called Quotes*Download*, the settings class is called Quotes*DownloadSettings* and the result class is called Quotes*Result*. `Base.DownloadClient<T>` class is abstract (MustInherit) so I will use `Finance.YahooFinance.QuotesDownload` as example.

Collapse | Copy Code

```
QuotesDownload dl = new QuotesDownload();
DownloadClient<QuotesResult> baseDl = dl;

QuotesDownloadSettings settings = dl.Settings;
settings.IDs = new string[] { "MSFT", "GOOG", "YHOO" };
settings.Properties = new QuoteProperty[] { QuoteProperty.Symbol,
                                            QuoteProperty.Name,
                                            QuoteProperty.LastTradePriceOnly
                                          };
SettingsBase baseSettings = baseDl.Settings;
```

In this example you need to set the IDs of the stocks you want to get data for. And you have to set the quote properties for the data rows. `SettingsBase` class only provides internal functions and outside the library it's only important for standardization.

Now you can start downloading the data. For that the `Base.DownloadClient<T>` class provides the two methods `Download` and `DownloadAsync`. Both are parameterless and `DownloadAsync` provides an optional parameter for user arguments. The `Download` funtion returns the generic `Base.Response<T>`.

Collapse | Copy Code

```
Response<QuotesResult> resp = baseDl.Download();
```

Every specialized download class like `QuotesDownload` can have overloaded `Download` and `DownloadAsync` methods. If you use the standard parameterless methods (`userArgs` is not counted as an parameter here) the class will use the `Settings` property object for downloading. With overloaded methods it could be that a new settings object will be created with aid of the passed parameters. Also the `Settings` property object could be cloned and only the passed parameters will be setted new to the cloned object.

`Base.Response<T>` stores the result `T` and the `Base.ConnectionInfo`. `Base.ConnectionInfo` class provides some information about download process like download size, needed time, success state or exceptions.

Collapse | Copy Code

```
ConnectionInfo connInfo = resp.Connection;
if (connInfo.State == ConnectionState.Success)
{
    QuotesResult result = resp.Result;
    //...
}
else
{
    Exception ex = connInfo.Exception;
    Debug.WriteLine(ex.Message);
}
```

If you want to start an asynchronous download it's nearly the same. You can start multiple download processes with a single instance of download class. `Base.DownloadClient<T>` provides two events that occurs when the download process completed. The first event is the generic `AsyncDownloadCompleted`.

Collapse | Copy Code

```
object userArgs = (double)1.5;
dl.AsyncDownloadCompleted += this.QuotesDownload_Completed;
dl.DownloadAsync(userArgs);

private void QuotesDownload_Completed(DownloadClient<QuotesResult> sender,
                                      DownloadCompletedEventArgs<QuotesResult> e)
{
    sender.AsyncDownloadCompleted -= this.QuotesDownload_Completed;

    object userArgs = e.UserArgs;
    double dbl = (double)userArgs;

    SettingsBase baseSettings = e.Settings;
    QuotesDownloadSettings settings = (QuotesDownloadSettings)baseSettings;

    Response<QuotesResult> resp = e.Response;
    QuotesResult result = resp.Result;
    //...
}
```

The delegate passes the `DownloadCompletedEventArgs<T>`. This class provides passed user arguments, used settings and the response with connection info and result data, etc. The settings will be cloned before the download started. So, if the settings of download object changed during async downloading, the settings of event args will have original values of current download process.

The second event is the non-generic `AsyncDownloadCompletedEvent` and is implemented by `IDownload` interface. `IDownload` is nearly as powerfull as `DownloadClient<T>`, without the generic part.

Collapse | Copy Code

```
object userArgs = (double)1.5;
dl.AsyncDownloadCompletedEvent += this.IDownload_Completed;
dl.DownloadAsync(userArgs);

private void IDownload_Completed(IDownload sender, IDownloadCompletedEventArgs e)
{
    sender.AsyncDownloadCompletedEvent -= this.IDownload_Completed;

    object userArgs = e.UserArgs;
    double dbl = (double)userArgs;

    SettingsBase baseSettings = e.Settings;
    QuotesDownloadSettings settings = (QuotesDownloadSettings)baseSettings;

    IResponse resp = e.GetResponse();
    object baseResult = resp.GetObjectResult();
    QuotesResult result = (QuotesResult)baseResult;
    //...
}
```

In this example I used the `Finance.YahooFinance.QuotesDownload` class. With help of the following overview you can see the most specialized download classes of this library.

- Finance.YahooFinance
  - `AlphabeticIDIndexDownload`: Downloads valid Yahoo! Finance IDs by index (A-Z).
  - `ChartDownload`: Downloads technical analyzing charts.
  - `CompanyInfoDownload`: Downloads main informations of companies.
  - `CompanyProfileDownload`: Downloads profile data of companies.
  - `CompanyStatisticsDownload`: Downloads financial statistics of companies.
  - `FuturesChainDownload`: Downloads future chains.
  - `HistQuotesDownload`: Downloads historic quotes.
  - `IDSearchDownload`: Searches for valid Yahoo! Finance IDs by keyword.
  - `MarketDownload`: Downloads market overview with sectors, industries and containing companies.
  - `MarketQuotesDownload`: Downloads the market quotes of a sector, industry or containing company.
  - `QuotesDownload`: Downloads up to 85 quote properties.
  - `QuoteOptionsDownload`: Downloads put and pull options.
- Finance.YahooPortfolio
  - `YPortfolioManager`: Manages your Yahoo! portfolios (create, edit and delete portfolios, views and components).
  - `HoldingsDownload`: Downloads holdings data.
  - `PortfolioInfoDownload`: Downloads an array of all portfolios without components.
  - `PortfolioDownload`: Downloads the components of a portfolio view.
- Finance.YahooScreener
  - `BondScreenerDownload`: Monitors bonds by specific criterias.
  - `StockScreenerDownload`: Monitors stocks by specific criterias
- Search.BOSS
  - `SearchDownload`: Downloads search query data for web, image, news and spelling service.
  - `RelatedSuggestionsDownload`: Downloads related search suggestions.
- Geo.GeoPlanet
  - `PlacesDownload`: Downloads geo data with Geo Planet API (deprecated by Yahoo!).
- Geo.PlaceFinder
  - `PlaceFinderDownload`: Downloads geo data with Place Finder API.
- Weather.YahooWeather
  - `LocationIDSearchDownload`: Downloads Location IDs for Weather API.
  - `WeatherFeedDownload`: Downloads weather forecast.

Additionally to these Yahoo! namespaces there is also the RSS namespace. Some web services provides RSS feeds and this class can be used to download RSS 2.0 feeds.

- RSS
  - `FeedDownload`: Downloads RSS 2.0 feeds.

This project started with wrapping Yahoo! web services but can also be extended for other web services. At this time there exists an implementation for MSN Money (Microsoft).

- Finance.MSNMoney

- ChartDownload: Downloads technical analyzing charts.
- HistQuotesDownload: Downloads historic quotes.
- IDSearchDownload: Searches for valid Yahoo! Finance IDs by keyword.
- QuotesDownload: Downloads quote properties.

Because of extension posibilities the namepsaces are structured like this: *MaasOne*.[*KindOfWebService*].[*NameOfWebService*]. So, it's possible to use synergies between same kinds of web services but different web service providers.

Finance.YahooFinance

Every example in this section needs the following imports:

Collapse | Copy Code

```
using MaasOne.Finance;
using MaasOne.Finance.YahooFinance;
```

The `Finance.YahooFinance` namespace contains the classes for Yahoo! Finance web service. The download principle is always the same, but I will show you some examples for easier understanding.

At first we want to get some IDs. For that there are different classes. One of them is `IDSearchDownload`.

Collapse | Copy Code

```
IDSearchDownload dl = new IDSearchDownload();
IDQuerySearchDownloadSettings settings = new IDQuerySearchDownloadSettings();
settings.Query = "dow";
settings.Markets = FinancialMarket.AllMarkets;
settings.ResultsIndex = 0;
settings.Server = YahooServer.Germany;
settings.Type = SecurityType.Any;
dl.Settings = settings;

Response<IDSearchResult> resp = dl.Download();

foreach (IDSearchData id in resp.Result.Items)
{
    string idStr = id.ID;
}
```

With `IDQuerySearchDownloadSettings` you can download maximum 20 results per request and you can set the start index if there are more than 20 results at all. With `IDInstantSearchDownloadSettings` you can download maximum 10 results without start index and no specifications like server, ranking or type etc.

A little bit special is the ID search by alphabetic index, because it's a combination of `AlphabeticIDIndexDownload` and `IDSearchDownload`.

Collapse | Copy Code

```
//Download TopIndex (e.g. [A], [B], [1-9])
AlphabeticIDIndexDownload dl = new AlphabeticIDIndexDownload();
dl.Settings.TopIndex = null;
Response<AlphabeticIDIndexResult> resp = dl.Download();

//Download Index (e.g. [C], [Cl], [Ca(1/10)])
AlphabeticalTopIndex topIndex = (AlphabeticalTopIndex)resp.Result.Items[2];
dl.Settings.TopIndex = topIndex;
Response<AlphabeticIDIndexResult> resp2 = dl.Download();

//Download ID Search Results
AlphabeticalIndex index = resp.Result.Items[0];
IDSearchDownload dl2 = new IDSearchDownload();
Response<IDSearchResult> resp3 = dl2.Download(index);

IDSearchData[] idResults = resp3.Result.Items;
```

With the received IDs it's possible to use the other classes for downloading data. I already showed one example in the "Download Classes" section above. Here another example with `HistQuotesDownload`.

Collapse | Copy Code

```
HistQuotesDownload dl = new HistQuotesDownload();
dl.Settings.IDs = new string[] { "GOOG" };
dl.Settings.FromDate = new DateTime(2010, 1, 1);
dl.Settings.ToDate = DateTime.Today;
dl.Settings.Interval = HistQuotesInterval.Weekly;

Response<HistQuotesResult> resp = dl.Download();

foreach (HistQuotesDataChain hqc in resp.Result.Chains)
{
    foreach (HistQuotesData hqd in hqc)
    {
        double close = hqd.Close;
        double closeAdj = hqd.CloseAdjusted;
        long volume = hqd.Volume;
        //...
    }
}
```

The other download classes in this namespace (and also most in this library) are working with the same principle: instanciate, set settings, download response, use result data.

Finance.YahooFinance.Support

Support namespace provides a lot of classes that are not directly necessary for getting the data. This namespace is just for handling with these data easier. For that, you have different data classes. The Support.YID class manages all information that are connecting to the Yahoo! Finance ID. Such an ID is constructed relatively structured. For getting data of the Apple stock, which is traded at XETRA stock exchange, you need the ID "AAPL.DE". The stock that is traded at NYSE has just the ID "AAPL". That shows that in most cases, you can deduce the stock exchange from the ID suffix (but there is more than one US American stock exchange, that has no suffix). For that issue, the YID class provides the StockExchange property for storing information like name, ID or trading time. You can prove if trading is active at a specific time in the machine's local timezone or UTC timezone. You can create a new instance of YID with an IDSearchResult. In most cases, the search result provides a known stock exchange and the YID constructor will manage the result data automatically. The Support.WorldMarket class manages all known, Yahoo! supported stock exchanges and a collection of some indices. You have a structure of continents, countries and indices to reflect the world market situation (provided by Yahoo!). A stock index will be represented by the YIndexID class that inherits from YID. Here you have the additional property DownloadComponents, which indicates if the downloader will load the data of the index itself or of the stocks of the index. Another IID class is YCurrencyID. This class represents currency relations of a base currency and a dependent currency. For that, there are the Currency properties BaseCurrency and DepCurrency. The result (QuotesDownload -> LastTradePriceOnly) you will download has the format 1 BaseCurrency : X DepCurrency.

Finance.YahooPortfolio

Every example in this section needs the following imports:

Collapse | Copy Code

```
using MaasOne.Finance.YahooPortfolio;
```

With the library you're able to manage your online portfolio. In general you are doing this with aid of YPortfolioManager class. This class provides several methods for creating, editing and deleting portfolios, views and components. YPortfolioManager inherits from YAccountManager, which provides the base methods for managing login status.

Collapse | Copy Code

```
private YAccountManager mManager = new YAccountManager();

private void LogIn()
{
    System.Net.NetworkCredential cred = new System.Net.NetworkCredential();
    cred.UserName = "username@yahoo.com";
    cred.Password = "password";
    bool isLoggeIn = mManager.LogIn(cred);
}
```

For downloading your portfolio overview you just need PortfolioInfoDownload class and YAccountManager.

Collapse | Copy Code

```
if (mManager.IsLoggedIn)
{
    PortfolioInfoDownload dl = new PortfolioInfoDownload();
    dl.Settings.Account = mManager;
    Response<PortfolioInfoResult> resp = dl.Download();
    foreach (PortfolioInfo pfi in resp.Result.Items)
    {
        string id = pfi.ID;
        string name = pfi.Name;
    }
}
```

If you want to download a special view of a single portfolio you need also YAccountManager and the specific portfolio ID for PortfolioDownload settings. Real-Time and Fundamentals view are static. They can not be edited or deleted.

Collapse | Copy Code

```
if (mManager.IsLoggedIn)
{
    PortfolioDownload dl = new PortfolioDownload();
    dl.Settings.Account = mManager;
    dl.Settings.PortfolioID = "your_portfolio_id";
    dl.Settings.ViewIndex = 0; //The first view
    //dl.Settings.DownloadRealTimeView = true; //Real-Time
    //dl.Settings.DownloadFundamentalsView = true; //Fundamentals

    Response<Portfolio> resp = dl.Download();

    Portfolio pf = resp.Result;

    string[] allViews = pf.AvailableViews;
    string selView = pf.SelectedView;

    foreach (IID id in pf.IDs)
    {
        string idStr = id.ID;
    }


    PortfolioColumnType[] columns = pf.Columns;
    foreach (PortfolioColumnType clm in columns)
    {
        Debug.Write(clm.ToString() + "|");
    }
    Debug.Write("\n");
    foreach (PortfolioDataRow row in pf.Rows)
    {
        foreach (PortfolioColumnType clm in row.AvailableColumns)
        {
            object cellValue = row[clm];
            Debug.Write(cellValue.ToString());
        }
        Debug.Write("\n");
    }
}
```

These examples were only simple downloading work. For editing your portfolio you need to upload the new data you want to change. For that you have the `YPortfolioManager`. This is not like the other download classes. You don't have a `Settings` property and you have several events for different actions.

Collapse | Copy Code

```
private YPortfolioManager mPfManager = new YPortfolioManager();
```

After logging in you can add a new portfolio.

Collapse | Copy Code

```
Response<Portfolio> createResp = mPfManager.CreatePortfolio("MyPortfolio");
Portfolio pf = createResp.Result;
Debug.WriteLine("New Portfolio: " + pf.Info.Name);
```

Now you could change the name.

Collapse | Copy Code

```
mPfManager.EditPortfolio(pf.Info.ID, "MyPortfolio_NewName");

Response<Portfolio> editResp = mPfManager.DownloadPortfolio(pf.Info.ID);
pf = editResp.Result;
Debug.WriteLine("Edit Name: " + pf.Info.Name);
```

Or delete it again.

Collapse | Copy Code

```
Response<PortfolioInfoResult> deleteResp = mPfManager.DeletePortfolio(pf.Info.ID);
PortfolioInfo[] restPortfolios = deleteResp.Result.Items;
```

The next step would be adding new stocks to the portfolio.

Collapse | Copy Code

```
Response<Portfolio> addResp = mPfManager.AddPortfolioItem("portfolioID", "GOOG");
Portfolio pf = addResp.Result;
foreach (IID id in pf.IDs)
{
    if (id.ID == "GOOG")
    {
        Debug.WriteLine("found");
    }
}
```

Of course you're also able to delete them.

Collapse | Copy Code

```
mPfManager.DeletePortfolioItem(pf.Info.ID, "GOOG");
```

In Yahoo! Portfolio you can set holdings for your stocks. At first you have to download the actual data. You can do that with `HoldingsDownload`/`YAccountManager` or `YPortfolioManager`.

Collapse | Copy Code

```
HoldingsDownload dl = new HoldingsDownload();
dl.Settings.Account = mManager;
dl.Settings.PortfolioID = "portfolioID";
Response<HoldingsResult> resp = dl.Download();

Holding[] holdings = resp.Result.Items;

foreach (Holding h in holdings)
{
    string id = h.ID;
    int shares = h.Shares;
    double pricePaid = h.PricePaid;
    Nullable<DateTime> tradeDate = h.TradeDate;
    //...
}
```

Now you can edit and update the values.

⊟ Collapse | Copy Code
```
holdings[0].PricePaid = 35.29;
holdings[0].Shares = 100;
holdings[0].TradeDate = DateTime.Today;

mPfManager.EditHoldings("portfolioID", holdings);
```

Hint: You can use `EditHoldings` method to add or delete multiple items/IDs of your portfolio or clear it completely (empty Array). The portfolio will be updated completely by the passed Holding-Array.

Creating, editing and deleting portfolio views is nearly the same, so I will jump now to Yahoo! Search BOSS.

Search.BOSS

Every example in this section needs the following imports:

⊟ Collapse | Copy Code
```
using MaasOne.Search.BOSS;
```

Yahoo! Search BOSS is the latest web service for using Yahoo!'s web, image and news search (also spelling and blogs). The library is using BOSS v2. If you want to use that service you have to register and create an OAuth key and to pay for it. After getting that OAuth key you're able to use this part of Yahoo! Managed to simply automate your BOSS queries.

For that issue you have the `Search.BOSS.SearchDownload` class. Primarily you have the normal `SearchDownloadSettings` class for setting OAuth credentials etc.

⊟ Collapse | Copy Code
```
SearchDownload dl = new SearchDownload();

SearchDownloadSettings settings = dl.Settings;
dl.Settings.ConsumerKey = "your_oauth_key";
dl.Settings.ConsumerSecret = "your_oauth_secret";
dl.Settings.HttpsUsed = true;
```

After setting universal options you have to set the service you want to use. It's possible to use multiple services in a single query.

⊟ Collapse | Copy Code
```
Culture culture = new Culture(Language.en, Country.US);
string queryText = "test";

SearchService service = null;

WebSearchService web = new WebSearchService();
web.LimitedWeb = true;
service = web;
service.Culture = culture;
service.Query = queryText;
service.Index = 0;
service.Count = 10;
dl.Settings.Services.Add(service);

NewsSearchService news = new NewsSearchService();
news.AlwaysLatestDateNow = true;
service = news;
service.Culture = culture;
service.Query = queryText;
service.Index = 0;
service.Count = 10;
dl.Settings.Services.Add(service);

ImageSearchService images = new ImageSearchService();
images.Dimensions = ImageSearchDimensions.All;
service = web;
service.Culture = culture;
service.Query = queryText;
service.Index = 0;
service.Count = 10;
dl.Settings.Services.Add(service);
```

Then you can start downloading the data. The result is splittet into different `SearchDataConatiner` classes for each service type. There you have the `Type` property which indicates the type of service and the type of specialized container class like `WebSearchDataContainer`. For the Items you have the `SearchData` class and it's specialized service inheritors like `WebSearchData`.

⊟ Collapse | Copy Code
```
Response<SearchResult> resp = dl.Download();

foreach (SearchDataContainer container in resp.Result.Containers)
{
    if (container.Type == SearchResultType.Web)
    {
        WebSearchDataContainer webContainer = (WebSearchDataContainer)container;
        foreach (WebSearchData wsd in webContainer.Items)
        {
            SearchData sd = wsd;
            string title = sd.Title;
            string description = sd.Description;
            Uri url = sd.Url;
            Uri clickUrl = sd.ClickUrl;

            string displyUrl = wsd.DisplayUrl;
            Language lang = wsd.Language;
            DateTime crawlDate = wsd.CrawlingDate;
        }
    }
}
```

Participate in the Project

I'm looking forward to extend this project to other web services. If you're also interested, you could participate in the project. The main work would be creating Settings classes (and URLs) and result parsing methods. The frame of downloading data is already available.

If you're interested just send a mail.

Thanks To

- Angelo Cresta for great bug fixing/testing work
- Zvonimir Digas for the stock exchanges timetable
- Alain Dionne for Canadian indices information

History

- 1st May, 2012
    - Version 0.11.2
        - `YPortfolioManager` changed
        - `YAccountManager` changed
        - `HistQuotesData` changed
- Version History from 0.7.8 to 0.11.2
- 26th November, 2010
    - Version 0.7.8
        - `QuotesDownload` internally changed
        - `HistQuoteData` changed
- 12th November, 2010
    - Version 0.7.7
        - Minor changes
        - Internal optimization
        - Miscellaneous bug fixes
        - Version for CF 3.5
- 15th September, 2010
    - Version 0.7.6
        - `Culture` class added
        - `Language enum` added
        - `Region enum` added
        - `Base.Download` fixed
        - `Support.YID` changed
        - `ChartDownload` extended
        - `FeedDownload` extended
        - `KeyStatisticsDownload` fixed
        - Internal optimization
        - Miscellaneous bug fixes
        - *market.xml* update
- 22nd July, 2010
    - Version 0.7.5
        - `Country enum` changed
        - `WorldMarket` changed
        - `CountryInfo` changed
        - `ContinentInfo` changed
        - `NonAPI.IDSearchDownload` changed
        - Miscellaneous bug fixes
        - *market.xml* update
- 21st July, 2010

- Version 0.7.4
    - `HistQuotesDownload` extended
    - `HistQuotesDataChain` added
    - `CompanyStatisticsDownload` fixed
    - `Base.Download` methods now `Protected` instead of `Friend`
    - `QuoteData.Values` changed
    - Internal optimization
    - Miscellaneous bug fixes

- 20th July, 2010
    - Version 0.7.3
        - `IDSearchDownloadChangedEventArgs` changed
        - `FinancialSecurityType` changed
        - `Feed` and `FeedDownload` changed
        - `Base.Download` methods now `Protected` instead of `Friend`
        - XML auto text encoding
        - Internal download structure changed
        - Internal optimization
        - Miscellaneous bug fixes
- 14th July, 2010
    - Version 0.7.2
        - `API.IDSearch` changed
        - `Base.ConnectionInfo` changed
        - Miscellaneous bug fixes
- 29th June, 2010
    - Version 0.7.1
        - `RSS.Feed` changed
        - Miscellaneous bug fixes
        - Internal optimization
        - *market.xml* update
- 24th June, 2010
    - Version 0.7
        - Download structure changed
        - `Threadsave` download
        - `Base.Response` added
        - `Base.ConnectionInfo` added
        - `CompanyStatisticsDownload` added
        - Namespace structure changed (subordination of Finance project)
        - `MarketDownload` changed
        - Miscellaneous bug fixes
        - Internal optimization
- 18th April, 2010
    - Version 0.6
        - `QuotesBaseData` added
        - `QuotesBaseDownload` added
        - `QuoteOption` added
        - `QuoteOptionsDownload` added
        - Data import/export changed
        - `StockExchange` changed
        - `MarketDownload` optimized
        - Miscellaneous bug fixes
        - Internal optimization
        - *market.xml* update
- 24th March, 2010
    - Version 0.5
        - Namespace structure changed
        - Version for Compact Framework 2.0
        - ID Search implemented
        - Alphabetical ID List download implemented
        - `ISIN` class added
        - Correction of `DownloadFailure` args (obsolete)
        - Data import/export changed
        - Minor bug fixes
        - Internal optimization
        - *market.xml* update

- 14th January, 2010
  - Version 0.4.3
    - YQL URL alteration
- 6th January, 2010
  - Version 0.4.2
    - Proxy support added
- 14th December, 2009
  - Version 0.4.1
    - Commodities added
    - Special currencies added
- 10th December, 2009
  - Version 0.4
    - YQL implementation
    - Data import/export
    - `StockExchange` added
    - `IID` interface added
    - `MarketDownload` added
    - Minor bug fixes
    - *market.xml* update
- 22nd November, 2009
  - Version 0.3
    - Currency exchange structure changed
    - `YID` added
    - Stock exchanges added
    - Downloader bug fix
    - `Servers` completed
- 27th October, 2009
  - Version 0.2
    - CSV reader bug fix
    - Clean disposing of download objects
    - `QuoteData` modified
    - Quote download of several indices
    - Indices added
    - `Currencies` completed
    - `Enum` description
- 26th September, 2009
  - Version 0.1
    - `Base.Download` added
    - Servers added
- 24th September, 2009
  - Updated source code
- 23rd September, 2009
  - Initial post

License

Share

EMAIL

About the Author

No Biography provided

MaasOne

Germany

Article Top

Comments and Discussions

Add a Comment or Question

Search Comments Go

Profile popups Spacing Relaxed Noise Medium Layout Normal Per page 25 Update

First Prev Next

Invalid Ticker Symbols	**ChuckSchu**	**22-Jul-14 12:39**
Quote decimal	**Alex160678**	**14-Jul-14 10:28**
My vote of 5	**SergeyWolf**	**5-Apr-14 15:42**
MaasOne.Finance.YahooFinance.AlphabeticIDIndexDownload() returning nothing	**testa16v**	**18-Dec-13 12:59**
Re: MaasOne.Finance.YahooFinance.AlphabeticIDIndexDownload() returning nothing	Member 4169628	2-Jan-14 2:59
Re: MaasOne.Finance.YahooFinance.AlphabeticIDIndexDownload() returning nothing	testa16v	21-Apr-14 11:21
Re: MaasOne.Finance.YahooFinance.AlphabeticIDIndexDownload() returning nothing	Ramajayam1989	5-Jan-14 0:43
Test Application Crash ...	**CreF**	**7-Aug-13 4:43**
QuotesDownload : Bug in CsvTextToStringTable(..)	**matildochka89**	**18-May-13 4:02**
Yahoo Finance does not provide some stock quotes	**samsantar**	**5-Apr-13 2:04**
Quotes test case	**GWSyZyGy**	**1-Mar-13 4:30**
Put it together	**Member 3927020**	**13-Feb-13 17:36**
Documentation on how to build or run this code?	**ydemissie**	**12-Feb-13 14:07**
StockScreenerDownload	**Robby Stamper**	**31-Jan-13 11:28**
Hello Maas	**Don G.**	**29-Dec-12 16:31**
waste of time	**nbrege**	**16-Dec-12 13:48**
Can't seem to run the app	**Soham Dasgupta**	**10-Oct-12 4:28**
I would like to contribute to this library	**ChriayuShah**	**15-Sep-12 19:05**
Source code for the Showcase project	**ljilekor**	**9-Sep-12 17:04**

newbie	**Member 9364149**	**17-Aug-12 7:24**
Still cant find next stocks	**billybond**	**10-Aug-12 21:10**
License for commercial application?	**tobogganracing**	**7-Aug-12 12:10**
Project Setup	**EG_3**	**11-Jul-12 7:06**
Where is the VB.net code?	**srarellano**	**28-Jun-12 10:05**
Re: Where is the VB.net code?	bradleykern	2-Mar-13 8:05

Last Visit: 13-Aug-14 19:53 Last Update: 20-Aug-14 7:41 Refresh 1 2 3 4 5 6 7 8 9 Next »

General News Suggestion Question Bug Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.