IBrokers

Automated Trading with R and Interactive Brokers

Jeffrey A. Ryan
jeffrey.ryan @ insightalgo.com

R/Finance 2010 Workshop

The Official API

www.interactivebrokers.com

Available in JAVA, C++, Excel, Visual Basic

Data, Orders, and Account Requests

The Official API

Exchanges

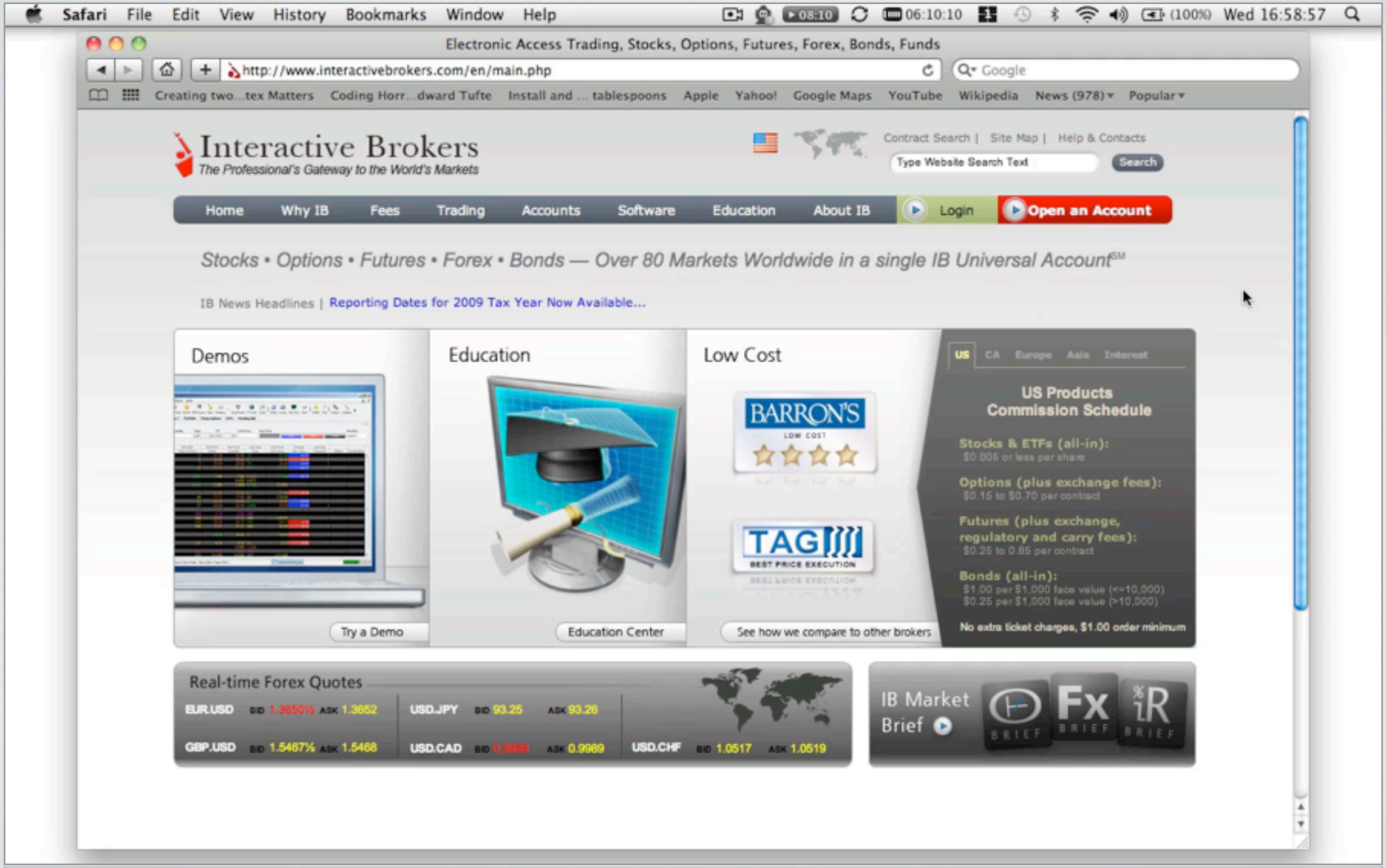
Toronto Stock Exchange
ISE
NYSE AMEX
CME
LAVA
ArcaEdge
BOX
EUREX
Hong Kong Stock Exchange
CBOE
London Stock Exchange
OneChicago
LIFFE
EURONEXT

TWS

Interactive Brokers **API**

The Official API

The Official API



The IBrokers API

Design and Motivation

Provide native R Access to IB API --- *no dependencies*

Keep official documentation *THE* documentation

Bring the power of R syntax into the equation

The IBrokers API

Connections

Contracts

Client Methods

The IBrokers API

Connections	Contracts	Client Methods
eWrapper	CALLBACK	processMsg

The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg

The IBrokers API

eWrapper + processMsg

The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

The IBrokers API

first, create a set of methods that do nothing with the incoming data and assign to eW

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

The IBrokers API

use eWrapper method assign.Data to create a list of xts objects to hold our bars -- one for each contract watched

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

The IBrokers API

create a new method realtimeBars to do what we want. In this case capture and print each new message in a format that makes sense

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

The IBrokers API

```
> eWrapper.RealTimeBars.CSV
function (n = 1)
{
    eW <- eWrapper(NULL)
    eW$assign.Data("data", rep(list(structure(.xts(matrix(rep(NA_real_,
        7), nc = 7), 0), .Dimnames = list(NULL, c("Open", "High",
        "Low", "Close", "Volume", "WAP", "Count")))), n))
    eW$realtimeBars <- function(curMsg, msg, timestamp, file,
        ...) {
        id <- as.numeric(msg[2])
        file <- file[[id]]
        data <- eW$get.Data("data")
        attr(data[[id]], "index") <- as.numeric(msg[3])
        nr.data <- NROW(data[[id]])
        cat(paste(msg[3], msg[4], msg[5], msg[6], msg[7], msg[8],
            msg[9], msg[10], sep = ","), "\n", file = file, append = TRUE)
        data[[id]][nr.data, 1:7] <- as.numeric(msg[4:10])
        eW$assign.Data("data", data)
        c(curMsg, msg)
    }
    return(eW)
}
<environment: namespace:IBrokers>
```

return the modified eWrapper object

The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg

The IBrokers API

Connections

Contracts

Client Methods

eWrapper

CALLBACK

processMsg

The IBrokers API

"snapshot" market data

The IBrokers API

"snapshot" market data

reqMktData snapshot=TRUE is *slow* and *wrong*

The IBrokers API

"snapshot" market data

reqMktData snapshot=TRUE is *slow* and *wrong*

Solution: re-implement with a custom CALLBACK

The IBrokers API

"snapshot" market data

reqMktData snapshot=TRUE is *slow* and *wrong*

Solution: re-implement with a custom CALLBACK

3 Lines of Code!

The IBrokers API

```
> snapShot
function (twsCon, eWrapper, timestamp, file, playback = 1, ...)
{
    if (missing(eWrapper))
        eWrapper <- eWrapper()
    names(eWrapper$.Data$data) <- eWrapper$.Data$symbols
    con <- twsCon[[1]]
        while (TRUE) {
            socketSelect(list(con), FALSE, NULL)
            curMsg <- .Internal(readBin(con, "character", 1L,
                NA_integer_, TRUE, FALSE))
            if (!is.null(timestamp)) {
                processMsg(curMsg, con, eWrapper, format(Sys.time(),
                  timestamp), file, ...)
            }
            else {
                processMsg(curMsg, con, eWrapper, timestamp,
                  file, ...)
            }
            if(!any(sapply(eWrapper$.Data$data, is.na)))
              return(do.call(rbind,lapply(eWrapper$.Data$data,as.data.frame)))
        }
}
```

The IBrokers API

```
> snapShot
function (twsCon, eWrapper, timestamp, file, playback = 1, ...)
{
    if (missing(eWrapper))
        eWrapper <- eWrapper()
    names(eWrapper$.Data$data) <- eWrapper$.Data$symbols
    con <- twsCon[[1]]
        while (TRUE) {
            socketSelect(list(con), FALSE, NULL)
            curMsg <- .Internal(readBin(con, "character", 1L,
                NA_integer_, TRUE, FALSE))
            if (!is.null(timestamp)) {
                processMsg(curMsg, con, eWrapper, format(Sys.time(),
                  timestamp), file, ...)
            }
            else {
                processMsg(curMsg, con, eWrapper, timestamp,
                  file, ...)
            }
            if(!any(sapply(eWrapper$.Data$data, is.na)))
              return(do.call(rbind,lapply(eWrapper$.Data$data,as.data.frame)))
        }
}
```

Putting it all together

Putting it all together

Extend CALLBACK and eWrapper objects
to implement a simple strategy:

Buy when ES crosses a specific threshold

Additional things to do...

Aggregate market data to OHLC bars with built in eWrapper function

Add filter interface to allow for CEP style queries

Complete the API:
fundamental data, scanner data

IBrokers

Automated Trading with R and Interactive Brokers

Thank You

Jeffrey A. Ryan
jeffrey.ryan @ insightalgo.com

R/Finance 2010 Workshop