

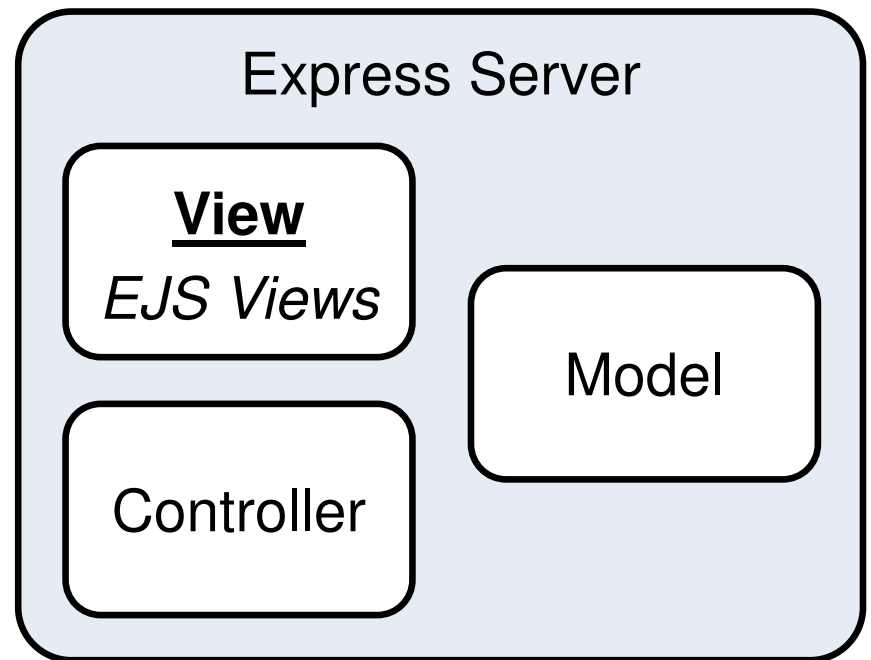
COMP4021  
Internet Computing

# MVC in Express Part 2

Gibson Lam

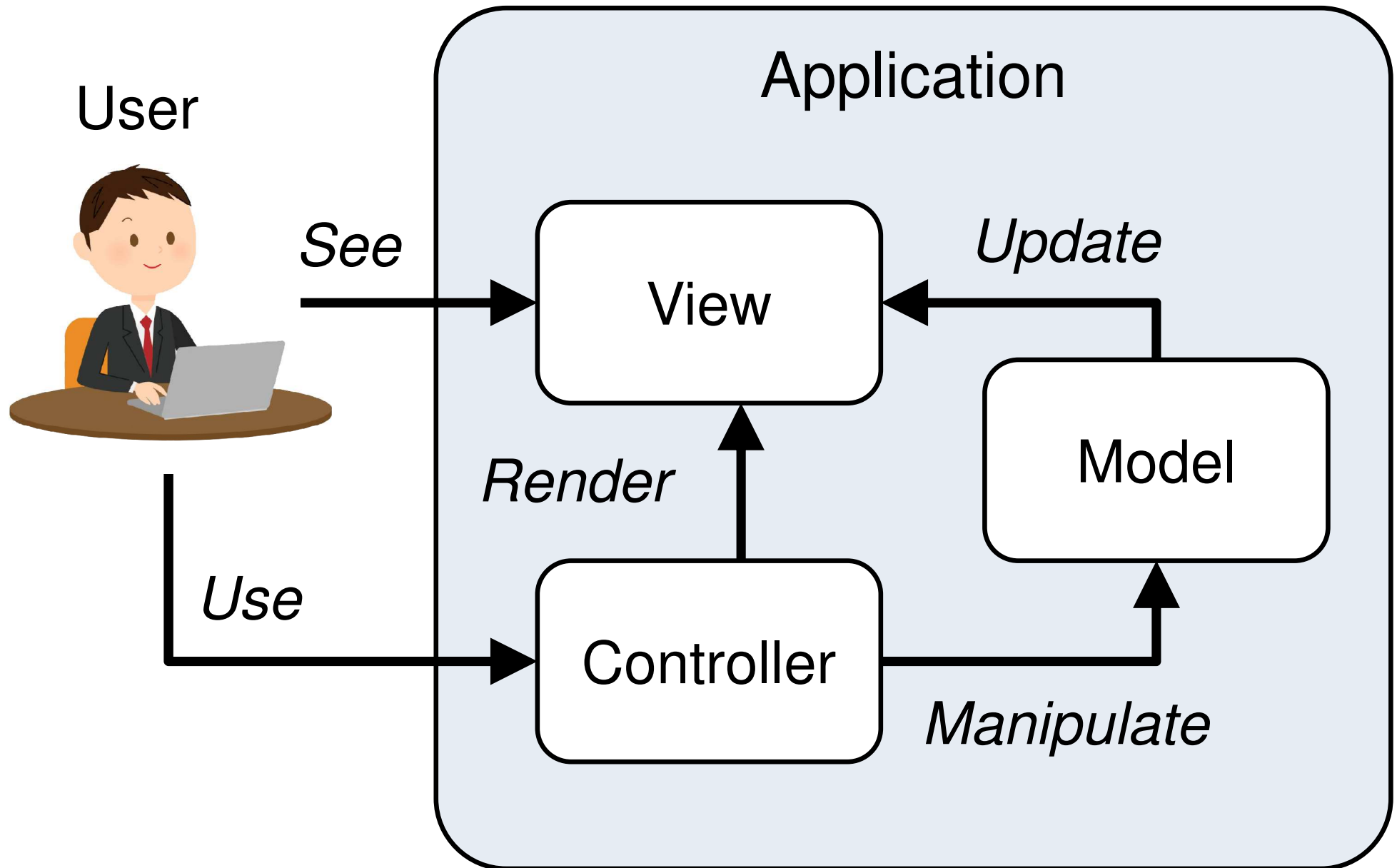
# This Presentation

- In the previous discussion, we talked about the MVC design
- We have used EJS in Express to create different views
- In this presentation we will continue with MVC building the models and controllers in Express



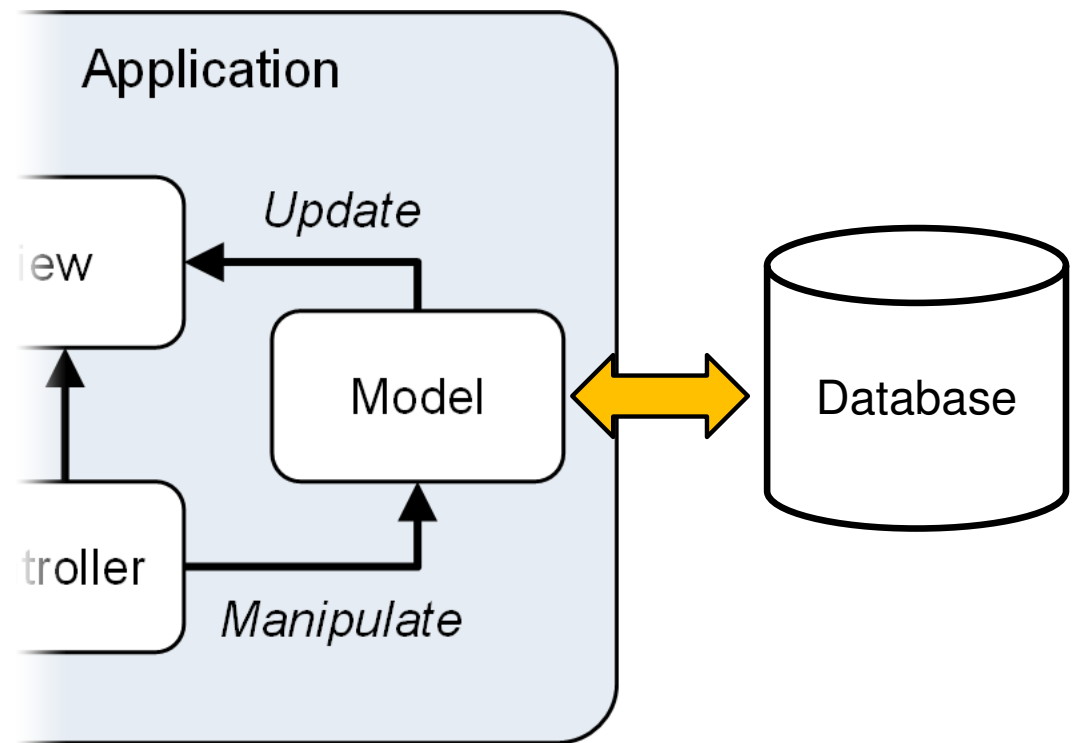
# Overview of MVC

You have seen  
this before



# Models in MVC

- Models in MVC manage all data-related functionalities
- Typically, they connect and retrieve data from databases located on some database servers

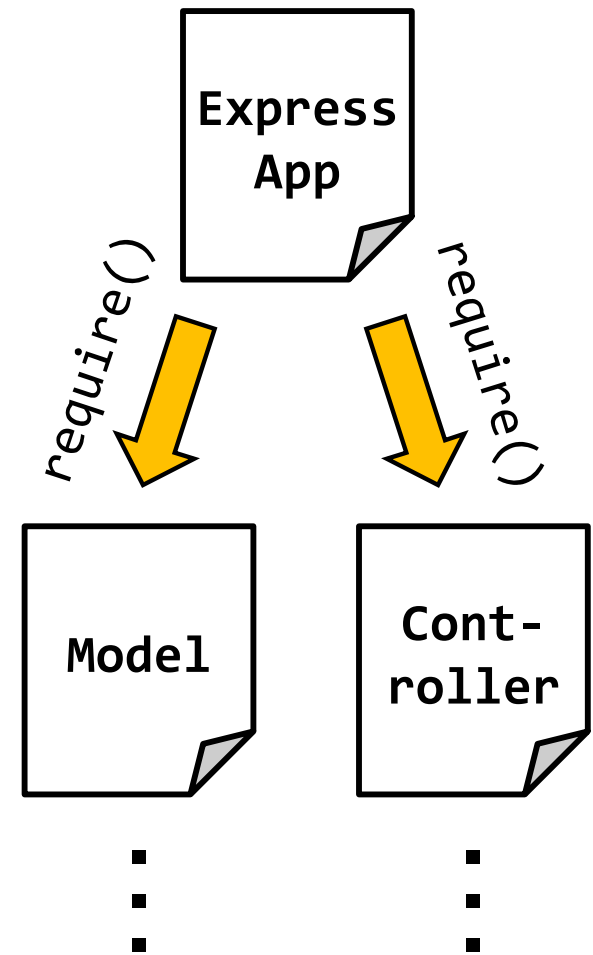


# Creating Models in Express

- Since we do not use any 'real' database, you can also create a model based on some data stored in, e.g.:
  - CSV files
  - JSON files
  - Remote locations/APIs
- In our example shown later in this discussion, we will use the data from a remote APIs

# Using Node.js Modules

- In an MVC Node.js project, you usually put models and controllers in separate files
- You can then use `require()` to import them when you need to
- Node.js modules work in a similar way to the JavaScript module patterns that you have used before



# The Exports Object

- In JavaScript module patterns, you create a module using a function
  - The function returns a JavaScript object, which contains the functionalities of the module
- In Node.js, you instead create a module using a global `module.exports` object
  - This object then contains the functionalities of the module and is implicitly returned by the `require()` command

# A Simple Module Pattern Example

```
const Counter = function() {  
    let count = 0;
```

*Increase  
and get a  
'counter'  
from the  
module*

```
    {  
        const increase = function() {  
            count = count + 1;  
        };  
        const get = function() {  
            return count;  
        };  
        return { increase, get };  
    };  
};
```



# Using the Module Pattern

- The module pattern in the previous slide can be created using this code:

```
const counter = Counter();
```

- Then, for example, you can increase the counter and get the counter back like this:

```
counter.increase();  
const value = counter.get();
```

# A Node.js Module Example

- A Node.js module with similar functionalities can be created in a file as shown below:

```
let count = 0;
```

```
const increase = function() {  
    count = count + 1;  
};
```

```
const get = function() {  
    return count;  
};
```

*Functions are  
exported by the  
exports object*

```
module.exports = { increase, get };
```



# Using the Node.js Module

- If the file in the previous slide is called `counter.js`, it can be imported by a Node.js program using this code:

```
const counter = require("./counter");
```

- The module can then be used in a similar way to its JavaScript counterpart, i.e.:

```
counter.increase();  
const value = counter.get();
```

# Making a Stock Model

- A stock data model can be created using a Node.js module
- The stock data is obtained using the Yahoo! Finance API
- A npm package called yahoo-finance2 lets us download Yahoo's stock data easily



# Using Yahoo! Finance

- To get stock data, you need to:

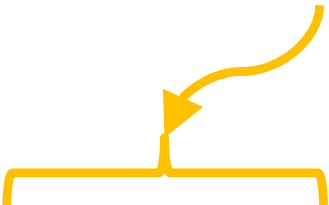
- Install the package:

```
>npm install yahoo-finance2
```

*You need  
to use  
.default  
here*

- Import the package:

```
const yahooFinance =  
    require('yahoo-finance2').default;
```



- Use either of these functions:

```
yahooFinance.quoteSummary() or  
yahooFinance.historical()
```

# Getting Stock Data

- For example, you can get the stock summary of Apple Inc using its stock code AAPL:

```
const summary =  
    yahooFinance.quoteSummary("AAPL");
```

- Or, get its historical stock prices using this code:

```
const prices =  
    yahooFinance.historical("AAPL",  
    { period1: "2001-01-01" });
```

# A Stock Model

- You can create the stock model like this:

```
const yahooFinance =  
  require('yahoo-finance2').default;  
  
async function getStock(code) {  
  const summary =  
    await yahooFinance.quoteSummary(code);  
  const prices =  
    await yahooFinance.historical(code,  
      { period1: "2001-01-01" });  
  
  return { summary, prices };  
}  
  
module.exports = { getStock };
```

*This is an async function*

*A single function is exported for getting the stock data*

# Using the Stock Model

- For example, you can use the stock model (`stocks.js`) to get stock data:

```
const stocks = require("./stocks");
```

```
stocks.getStock("AAPL")
```

```
.then((stock) => {
```

```
    ...Do something with the stock data...
```

```
})
```

```
.catch((error) => {
```

```
    console.log(error.message);
```

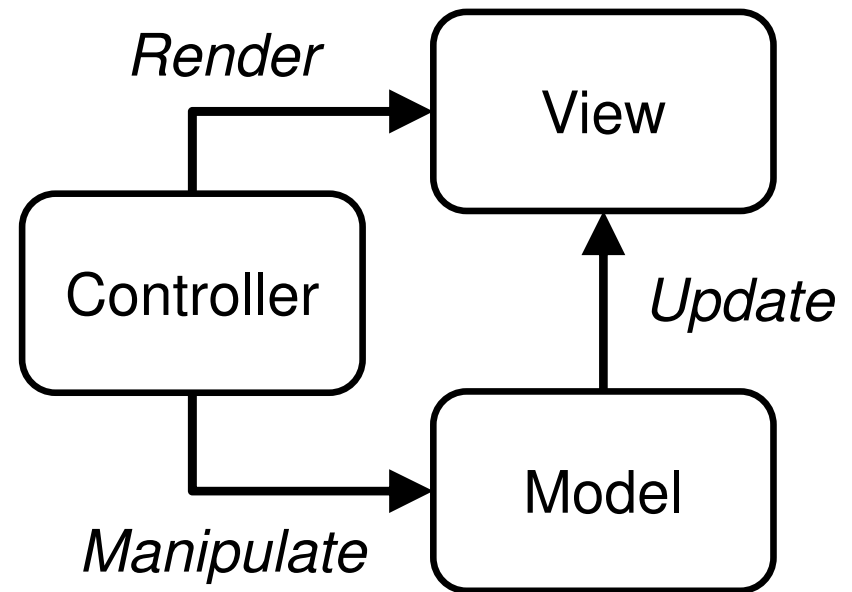
```
});
```

*You  
need to  
use a  
promise  
here*



# Controllers in MVC

- Controllers handle the requests from the users, load the models and pair the models with the appropriate views
- These look exactly like what the routes in Express, i.e. `app.get()` and `app.post()`, can do



# Making Controller Modules

- You can group the route callback functions of the same controller into a single Node.js module
- For example, here is part of the content of an 'authentication' controller:

```
const signIn = (req, res) => { ... };
const signOut = (req, res) => { ... };
...

module.exports = { signIn, signOut, ... };
```

# Importing the Controllers

- If the module in the previous slide is called `authentication`, you can set up the Express server like this:

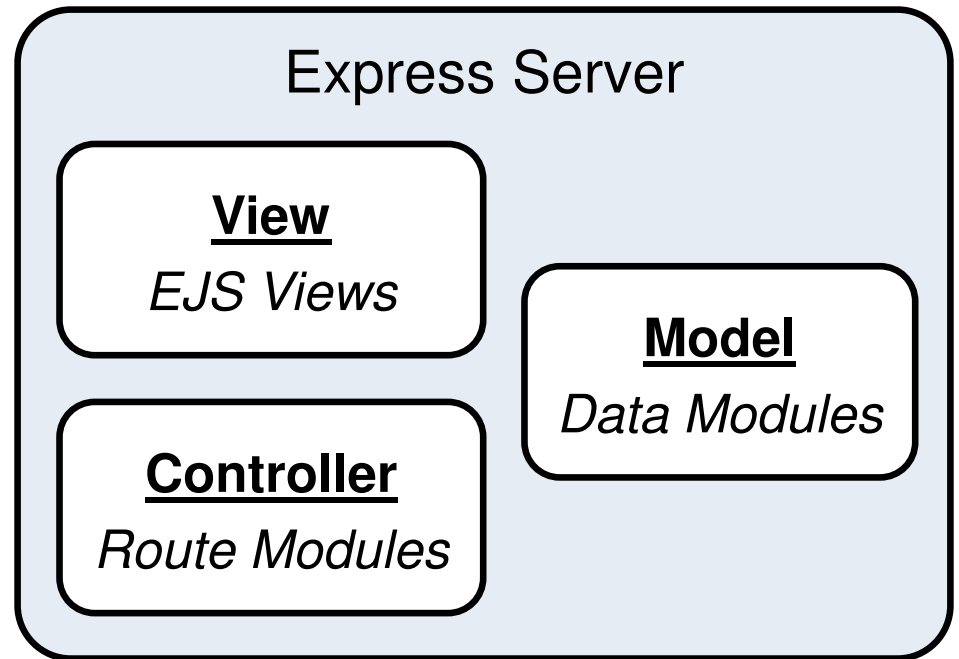
```
const authController =  
    require("./authentication");  
  
...  
app.post("/signin", authController.signIn);  
app.get("/signout", authController.signOut);
```



*These two endpoints are handled by  
the authentication controller*

# Using MVC in Express

- Based on what we have discussed so far, we can build an example web application using Express
- The application is a stock display system, where you can show the historical stock prices and the corresponding chart of a particular stock



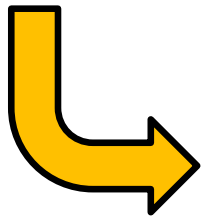
# The Stock Application

*Enter the stock code*

## Stock System

Stock Code:

Show



*See the  
stock prices*

### FB - Meta Platforms, Inc.

Table View

Show 10 entries

Date	Open	High	Low	Adjusted Close (\$)	Volume
3/31/2022	\$228.45	\$228.49	\$222.26	\$222.30	24192300
3/30/2022	\$228.91	\$231.15	\$226.71	\$227.85	25588000
3/29/2022	\$226.07	\$230.89	\$225.29	\$229.86	31417900
3/28/2022	\$222.13	\$224.04	\$219.54	\$223.59	26224100
3/25/2022	\$220.51	\$225.50	\$218.89	\$221.82	40039000
3/24/2022	\$215.00	\$220.67	\$214.79	\$219.57	31502300
3/23/2022	\$213.33	\$216.80	\$212.16	\$213.46	23717300
3/22/2022	\$211.37	\$219.46	\$210.18	\$216.65	31998800
3/21/2022	\$214.50	\$214.71	\$207.63	\$211.49	30142300
3/18/2022	\$206.70	\$216.80	\$206.00	\$216.49	52128000

Showing 1 to 10 of 2,484 entries

Previous

1

2

3

4

5

...

249

Next

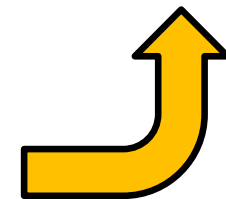
### FB - Meta Platforms, Inc.

Change Stock

Table View

Chart View

↑ Adjusted Close (\$)



*Or, see the  
stock chart*

# MVC in the Example

- Here are the components in the stock application:
  - 5 server endpoints
    - ‘/’, ‘/get’, ‘/change’, ‘/table’ and ‘/chart’
  - 1 model
    - The stocks model
  - 3 views
    - The welcome view, the stock table view and the stock chart view
  - 2 controllers
    - The front page controller and the stock controller

# The Server Endpoints 1/2

- The '/' endpoint
  - Show the welcome page using the welcome view
- The '/get' endpoint
  - Load the stock into the session storage
  - Redirect the browser to '/table'
  - This does not have a view
- The '/change' endpoint
  - Remove the stock from the session storage
  - Redirect the browser to '/'
  - This does not have a view

## Stock System

Stock Code:

# The Server Endpoints 2/2

- The `‘/table’` endpoint
  - Show the table view of the stock
  - Redirect the browser to `‘/’` if the stock is not loaded
- The `‘/chart’` endpoint
  - Show the chart view of the stock
  - Redirect the browser to `‘/’` if the stock is not loaded

FB - Meta Platforms, Inc. [Change Stock](#)

Table View Chart View

Show 10 entries

Date	Open	High	Low	Close	Adjusted Close	Volume
3/31/2022	\$228.45	\$228.49	\$222.26	\$222.36	\$222.36	24192300
3/30/2022	\$228.91	\$231.15	\$226.71	\$227.85	\$227.85	25588000
3/29/2022	\$226.07	\$230.89	\$225.29	\$229.86	\$229.86	31417900
3/28/2022	\$222.13	\$224.04	\$219.54	\$223.59	\$223.59	26224100
3/25/2022	\$220.51	\$225.50	\$218.89	\$221.82	\$221.82	40039000
3/24/2022	\$215.00	\$220.67	\$214.79	\$219.57	\$219.57	31502300
3/23/2022	\$213.33	\$216.80	\$212.16	\$213.46	\$213.46	23717300
3/22/2022	\$211.37	\$219.46	\$210.18	\$216.65	\$216.65	31998800
3/21/2022	\$214.50	\$214.71	\$207.63	\$211.49	\$211.49	30142300
3/18/2022	\$206.70	\$216.80	\$206.00	\$216.49	\$216.49	52128000

Showing 1 to 10 of 2,484 entries [Previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) ... [249](#) [Next](#)





# The Welcome View

- The welcome view is a plain HTML page
- If you enter a stock code and press 'Show', the form will be submitted to the '/get' endpoint
- If no error occurs, you will get to the stock table view page
- Example stock codes are AAPL, FB, 0001.HK and 0388.HK

## Stock System

Stock Code:

Show

# The Stock Table View

- The stock table view puts the stock prices in an HTML table
- The table is then nicely formatted using the Datatable JavaScript library
- You can click on 'Chart View' to see the chart (using '/chart') or 'Change Stock' to switch to another stock (using '/change')

0388.HK - Hong Kong Exchanges and Clearing Limited [Change Stock](#)

Table View [Chart View](#)

Show 10 entries

Date	Open	High	Low	Close	Adjusted Close	Volume
3/31/2022	HK\$376.00	HK\$376.20	HK\$369.80	HK\$371.40	HK\$371.40	4147679
3/30/2022	HK\$376.00	HK\$383.80	HK\$372.40	HK\$376.20	HK\$376.20	5173799
3/29/2022	HK\$375.20	HK\$376.80	HK\$370.00	HK\$372.00	HK\$372.00	3072303
3/28/2022	HK\$370.40	HK\$377.60	HK\$367.00	HK\$372.40	HK\$372.40	3839918
3/25/2022	HK\$380.00	HK\$385.00	HK\$370.80	HK\$373.60	HK\$373.60	4984150
3/24/2022	HK\$380.40	HK\$386.60	HK\$378.40	HK\$384.00	HK\$384.00	4285958
3/23/2022	HK\$378.00	HK\$388.00	HK\$378.00	HK\$382.80	HK\$382.80	5421773
3/22/2022	HK\$376.40	HK\$381.00	HK\$371.80	HK\$380.00	HK\$380.00	4236957
3/21/2022	HK\$382.80	HK\$383.00	HK\$373.40	HK\$374.20	HK\$374.20	3817968
3/18/2022	HK\$381.60	HK\$385.20	HK\$373.00	HK\$378.60	HK\$378.60	6012437

Showing 1 to 10 of 5,251 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [526](#) Next

# The Stock Chart View

- The stock chart view shows the chart of a period of the stock
- You can adjust the period using the slider at the top of the chart, which is drawn using the D3 JavaScript library
- You can switch to see the table view or another stock, similar to the table view



# The Controllers

- The front page controller manages only the welcome page of the system, i.e. the '/' endpoint
- The stock controller maintains all other endpoints, which manipulate the stock that you can see from the browser
- Using the controllers, the Express app can be set up concisely, as the application logic is inside the controllers

# Summary

- Using MVC, you can separate your code into models, views and controllers
- This allows you to focus on one aspect of the project, rather than mixing them all together
- It is not only useful in Express, but also in other client-side or server-side frameworks