THE UNIVERSITY OF SYDNEY

School of Computer Science

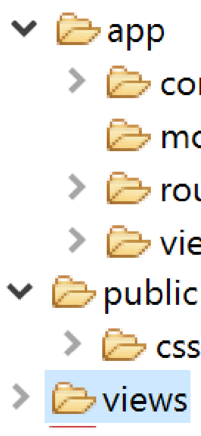
# COMP5347: Web Application Development Week 6 Tutorial: MVC Architecture and Sessions (Node.js and Express.JS Applications)

## Learning Objectives

- Build an application with separate controller and router
- Understand and practice session

## Part 1: Prepare folder structure

**Task:** Start Eclipse and open the project you have created from last week or use the provided solution from Week 5. Create new folders with a structure similar to Figure 1. Note that the "views" folder parallel to the "app" folder is the one you created in week 5. You can leave it there as is.



```
app
    controllers
    models
    routes
    views
public
    css
views
```

Figure 1: MVC based folder structure

## Part 2: Install express-session Package

Express has many modules that implement common functions for web application development. One of these modules is express-session which provides functions to manage sessions in Web applications. To use this module, you suppose to download it (“express-session” module).

### Managing Packages with Package.json

Before installing the express-session module, it is important to understand how to manage dependencies in Node.js projects.

*package.json* is a file that lists information about packages that your project depends on. You should specify the name and version of any package you would like to use in your project (e.g., express-session) to make your build reproducible and hence easy to share with other developers. The name and version must adhere to the semantic versioning rules[1].

A node.js project can be executed without “package.json” like what we did in the previous tutorial session. The “package.json” play the role of recording installation of each 3rd party module. If you share the project with the world, you do not have to provide the entire node_module folder, but you need to specify these packages/modules in the “package.json” file. The “node_modules” folder can be restored with the npm command “npm install” to install dependencies needed to run your project[2]. To align with the best practice, you are suggested to use package.json with your projects.

**Task:** To install express-session package, follow the one of the solutions below:

- Solution 1**:** Download the “node_modules_W6.zip” from Week 6 Module of COMP5347 Canvas site and extract the content from it. Then remove the node_modules we used in week 5 tutorial from your Week 5’s project folder. Then copy the “node_modules” inside “Node_modules_W6” into Week 5’s project folder – the new directory node_module_W6 contains the express-session module that we have prepared for you. Then copy the “package.json” to the folder as well (optional) – we have added the express-session package/module to this file.

- Solution 2: Run the command “npm install express-session --save” from your project’s root directory in a command line (CMD or terminal) to install the session package.

[1] https://docs.npmjs.com/getting-started/using-a-package.json
[2] https://docs.npmjs.com/cli/install

- Solution 3: If you already have a “package.json”, you can add the following line at the end of the “dependencies”

  ```
  "express-session": "^1.15.6"
  ```

  Save the file and right click it on the project explorer (on Eclipse) panel to “Run As” then “npm update”. Optionally, you can also run it using cmd/terminal by running “npm install” and “npm update” in your project’s root directory.

## Part 3: Convert week 5 survey app to full MVC structure

**Task:** In the following steps, we use the sample code provided inside “week6.zip” from Canvas to convert week 5 application to full MVC structure. Make sure you put the files in correct location and update their location reference accordingly in the code.

a) Copy “*survey.server.controller.js”* into folder “*app/controllers”*. You should read the comments and try to understand the code.

b) Copy “*survey.server.routes.js”* into folder “*app/routes”*, then investigate the code referring to “*survey.server.controller.js”*.

c) Copy *server.js* in your project’s **root** directory (nodejs-labs). This is the entry file that set up various configuration and start the server which is similar to week 6’s.

d) Move all files from “views” directory you used last week into folder *app/views*; afterwards remove “views” directory from the root directory (nodejs-labs) and keep “views” folder inside “app” folder.

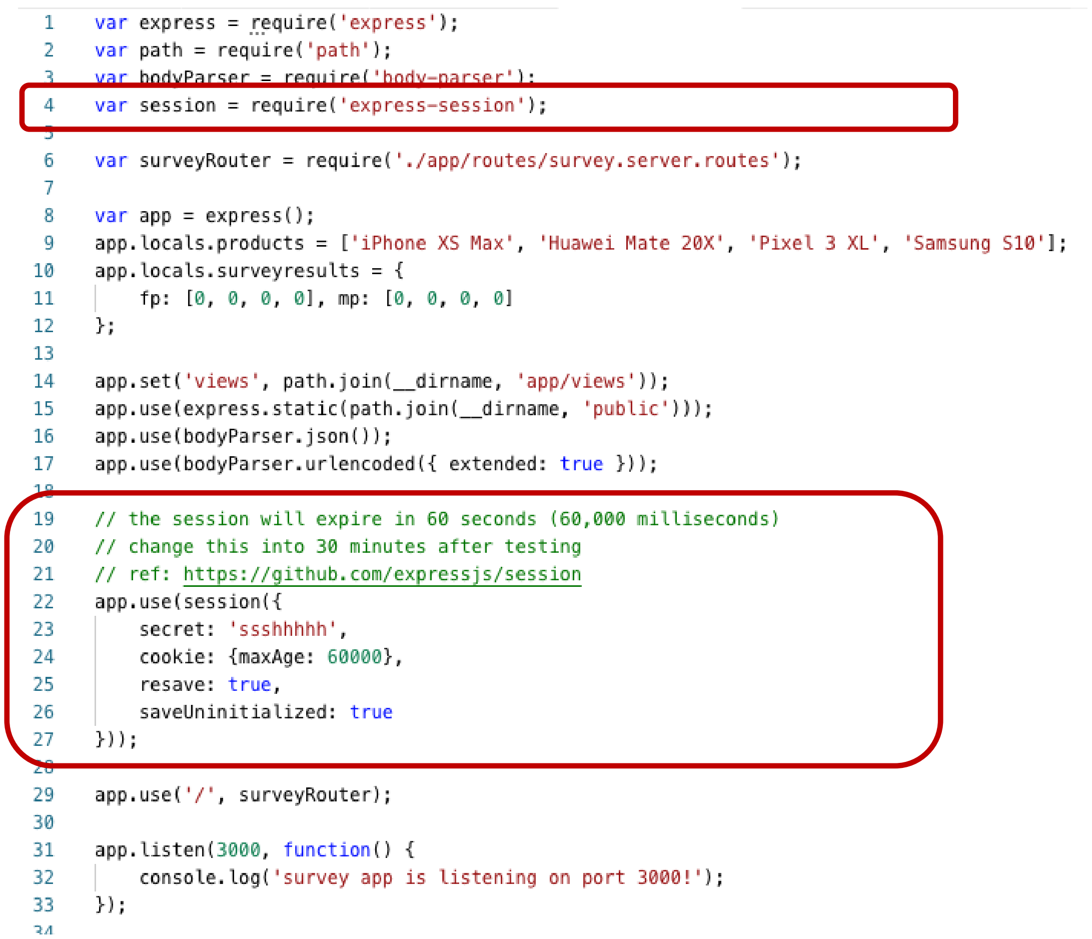
e) Now right click *server.js* on the project explorer panel (on Eclipse) and select from the dropdown menu to “Run As” “Node.js application”. If you are not using Eclipse, you can run command “*node server.js*” from terminal/cmd after navigating into your project’s root directory.

f) Navigate to “http://localhost:3000” to check the outcome with a browser.

## Part 4: Session Aware Survey

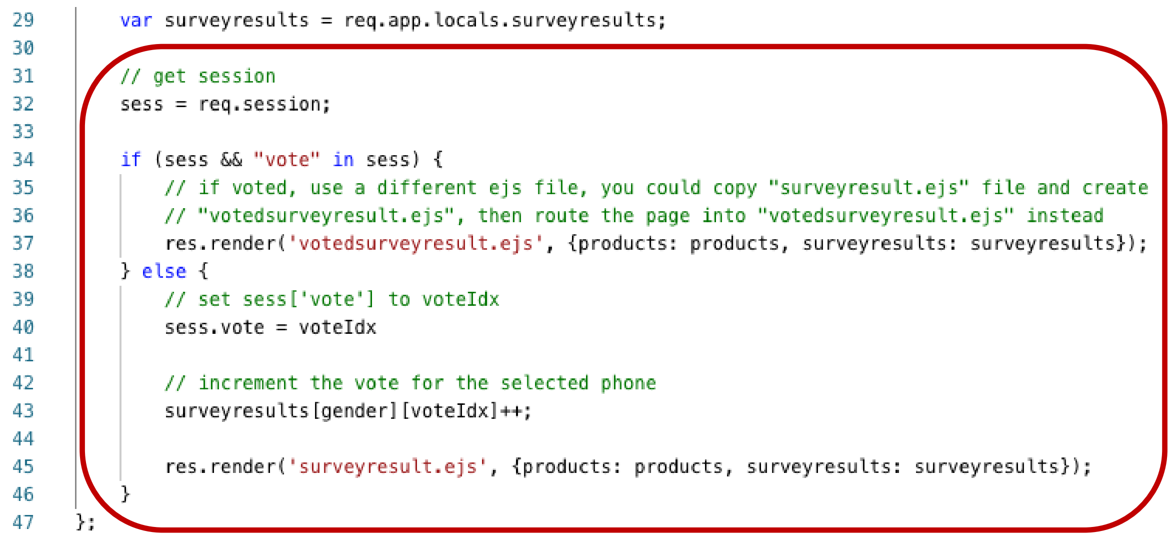
a) Figures below show sample code of a simple session aware survey that will prevent users from voting twice in 1-minute interval using the same browser (for development and testing purpose). You must change it to 60 minutes interval after the testing finishes.

   **Task:** Implement the code in your project and try to use chrome DevTools to inspect the cookies in various request and response header.



```
var express = require('express');
var path = require('path');
var bodyParser = require('body-parser');
var session = require('express-session');

var surveyRouter = require('./app/routes/survey.server.routes');

var app = express();
app.locals.products = ['iPhone XS Max', 'Huawei Mate 20X', 'Pixel 3 XL', 'Samsung S10'];
app.locals.surveyresults = {
    fp: [0, 0, 0, 0], mp: [0, 0, 0, 0]
};

app.set('views', path.join(__dirname, 'app/views'));
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

// the session will expire in 60 seconds (60,000 milliseconds)
// change this into 30 minutes after testing
// ref: https://github.com/expressjs/session
app.use(session({
    secret: 'ssshhhhh',
    cookie: {maxAge: 60000},
    resave: true,
    saveUninitialized: true
}));

app.use('/', surveyRouter);

app.listen(3000, function() {
    console.log('survey app is listening on port 3000!');
});
```

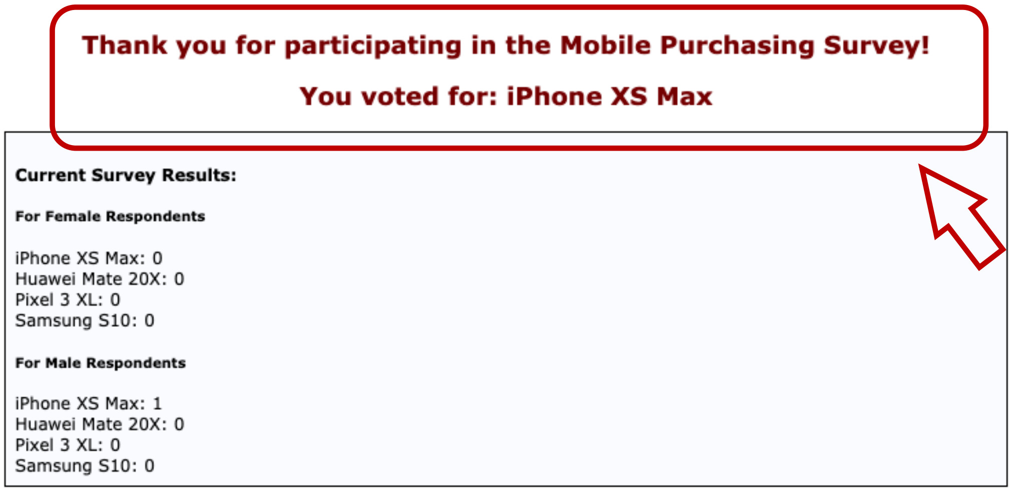
Figure 3: New server.js code to have session



```
    var surveyresults = req.app.locals.surveyresults;

    // get session
    sess = req.session;

    if (sess && "vote" in sess) {
        // if voted, use a different ejs file, you could copy "surveyresult.ejs" file and create
        // "votedsurveyresult.ejs", then route the page into "votedsurveyresult.ejs" instead
        res.render('votedsurveyresult.ejs', {products: products, surveyresults: surveyresults});
    } else {
        // set sess['vote'] to voteIdx
        sess.vote = voteIdx

        // increment the vote for the selected phone
        surveyresults[gender][voteIdx]++;

        res.render('surveyresult.ejs', {products: products, surveyresults: surveyresults});
    }
};
```

Figure 4: Additional lines in “survey.server.controller.js” to support session

b) **Task:** Now update the sample code to display more information in the result. In particular, if a user has not voted before, the result page should look like Figure 5. Most of the change happens in the result template where you need to add respective message depends on the session data. In the controller, you also need to pass the user's actual vote to the result template. If a user has voted, the result page should look like Figure 6.



**Thank you for participating in the Mobile Purchasing Survey!**

**You voted for: iPhone XS Max**

**Current Survey Results:**

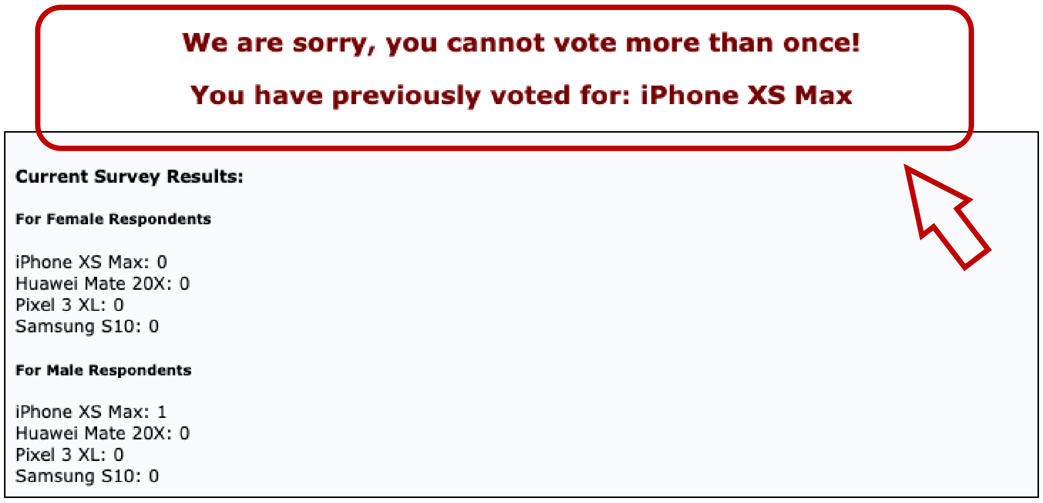
**For Female Respondents**

iPhone XS Max: 0
Huawei Mate 20X: 0
Pixel 3 XL: 0
Samsung S10: 0

**For Male Respondents**

iPhone XS Max: 1
Huawei Mate 20X: 0
Pixel 3 XL: 0
Samsung S10: 0

Figure 5: Result page for users who have not voted before



**We are sorry, you cannot vote more than once!**

**You have previously voted for: iPhone XS Max**

**Current Survey Results:**

**For Female Respondents**

iPhone XS Max: 0
Huawei Mate 20X: 0
Pixel 3 XL: 0
Samsung S10: 0

**For Male Respondents**

iPhone XS Max: 1
Huawei Mate 20X: 0
Pixel 3 XL: 0
Samsung S10: 0

Figure 6: Result page for users who have voted