School of Information Technologies

COMP5347: Web Application Development
Week 7 Tutorial: MVC Architecture and Sessions (Node.js/Express.js Application)

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 in week 6. Create the folder structure similar to Figure 1. Note that the "views" folder parallel to the "app" folder is the one you created in week 6. You can leave it there as is.

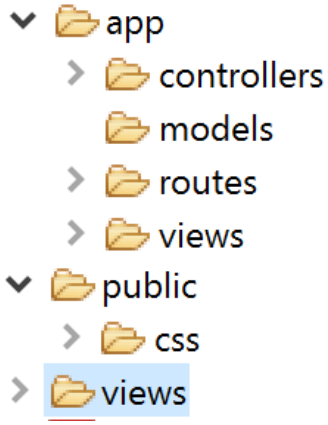


Figure 1: MVC based folder structure

Part 2: Install express-session Package

Express has got 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 (depending which machine you're using):

- **Solution 1 (Recommended):** Download the "node_modules_W7.zip" from Week 7 Module of COMP5347 Canvas site and extract the content from it. Then remove the node_modules we used in week 6 tutorial from your Week 6's prject folder. Then copy the new node_module_W7 in Week 6's project folder – the new directory node_module_W7 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" from your project's root directory in a commandline (CMD or PowerShell) 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": "^1.15.0"

  Save the file and right click it on the project explorer panel to Run As then npm update.

Part 3: Convert week 6 app to full MVC structure

Task: In the following steps, we use the sample code in Appendix A to convert week 6 application to full MVC structure. Make sure you put the files in correct location and update their location reference accordingly in the code.

a) Create a JavaScript file *survey.server.controller.js* in folder *app/controllers*.

b) Create a JavaScript file *survey.server.routes.js* in folder *app/routes*.

c) Create a JavaScript file *server.js* in your project’s **root** directory. This is the entry file that set up various configuration and start the server.

d) Copy the “*survey.pug*” you used last week to folder *app/views*; make sure you update the form’s action to the correct URL. Copy “*surveyresult.pug*” from week 6 solution to folder *app/views*

e) Now right click *server.js* on the project explorer panel and select from the dropdown menu to ‘Run As” “Node.js application”

Part 4: Session Aware Survey

a) Appendix B show sample code of a simple session aware survey that will prevent users from voting twice in 30 minutes interval using the same browser.

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

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 2. 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 3

Figure 2: Result page for users who have not voted before

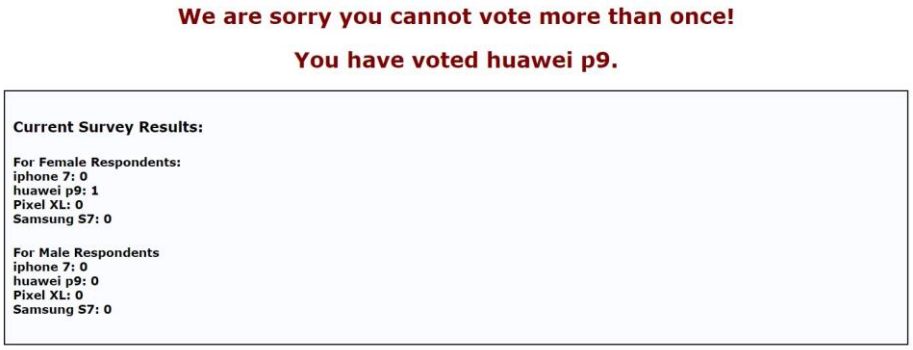
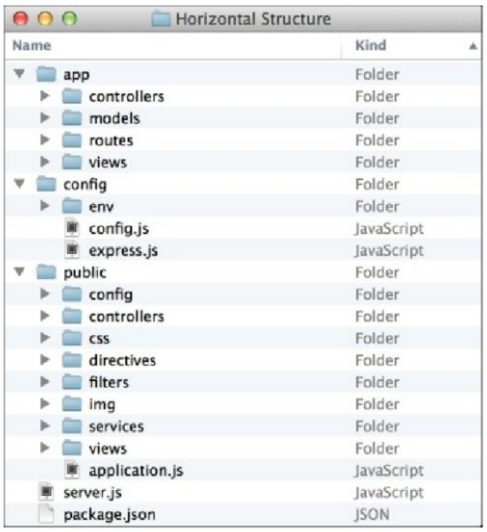


Figure 3: Result page for users who have voted

Appendix A



survey.server.controller.js

```
var express = require('express')

module.exports.showForm=function(req,res){
    products = req.app.locals.products
    res.render('survey.pug',{products:products})
}

module.exports.showResult=function(req,res){
    console.log(req.body);
    gender = req.body.gender
    productidx = req.body.vote;
    products = req.app.locals.products;
    surveyresults = req.app.locals.surveyresults;
    if (gender == 0)
        surveyresults.mp[productidx]++;
    else
        surveyresults.fp[productidx]++;
    res.render('surveyresult.pug', {products: products,
         surveyresults: surveyresults})
}
```

This controller module exposes two methods: **showForm** is used for displaying the form; **showReulsult** is used for showing the results

The methods are not mapped to URL yet

`req.app.locals` is used to share application scope variables

Each request object has a reference to the current running express application: **req.app**

`app.locals` is used to store properties that are local variables within the application (application scope data)

survey.server.routes.js

```
var express = require('express')
var controller = require('../controllers/survey.server.controller')
var router = express.Router()

router.get('/', controller.showForm)
router.post('/survey', controller.showResult)
module.exports = router
```

server.js

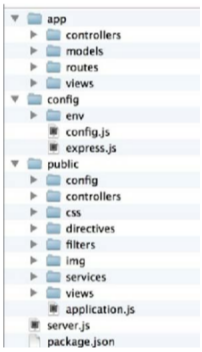
```
var express = require('express');
var path = require('path')
var bodyParser = require('body-parser');

var survey = require('../routes/survey.server.routes')

var app = express()
app.locals.products=['iphone 7', 'huawei p9', 'Pixel XL', 'Samsung S7']
app.locals.surveyresults = {
    fp:[0,0,0,0], mp:[0,0,0,0]
}

app.set('views', path.join(__dirname,'views'));
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.json())
app.use(bodyParser.urlencoded())
app.use('/survey', survey)
app.listen(3000, function () {
  console.log('survey app listening on port 3000!')
})
```

Set the two application scope variables



Get request send to /survey will display the form
Post request send to /survey/survey will show the result

Appendix B

server.js

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

var surveysession = require('./routes/surveysession.server.routes')

var app = express()
app.locals.products=['iphone 7', 'huawei p9', 'Pixel XL', 'Samsung S7']
app.locals.surveyresults = {fp:[0,0,0,0],mp:[0,0,0,0]}

app.set('views', path.join(__dirname,'views'));
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.json())
app.use(bodyParser.urlencoded())
app.use(session({secret: 'ssshhhhh',cookie:{maxAge:600000}}));
app.use('/session',surveysession)
app.listen(3000, function () {
  console.log('survey app listening on port 3000!')
})
```

The session will expire in 30 minutes

surveysession.server.controller.js

```
var express = require('express')

module.exports.showForm=function(req,res){
    products = req.app.locals.products
    res.render('surveysession.pug',{products:products})
}

module.exports.showResult=function(req,res){
    gender = req.body.gender
    productidx = req.body.vote;
    products = req.app.locals.products;
    surveyresults = req.app.locals.surveyresults;
    sess=req.session;
    if ("vote" in sess)
        res.render('surveysessionresult.pug', {products: products, surveyresults: surveyresults})
    else{
        sess.vote = productidx;
        gender = req.body.gender
        productidx = req.body.vote;
        if (gender == 0)
            surveyresults.mp[productidx]++;
        else
            surveyresults.fp[productidx]++;
        res.render('surveysessionresult.pug', {products: products, surveyresults: surveyresults})
    }
}
```