Lab 3 Part 1

Authorisation

Overview

In this lab, we will look at:

- Session management in Node.js and Express with the **express-session** module
- Access control functionality: Limit access to some routes in our web application for logged-in users only.

Tasks

Task 1: Prepare your environment

1. If you are working on your local machine (highly recommended!) you can continue to the next task.

2. If you are working directly on the virtual server (not recommended!) you will need to stop your previous application in order to complete this lab.

```
forever stopall
```

If your previous lab submission has not yet been marked, don't forget to start it again when you finish this lab. Otherwise we will not be able to mark your work!

This week's lab is not assessed this week.

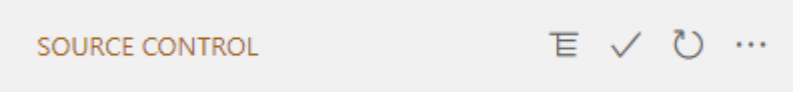
Task 2: Tag your work from Lab 2 (optional)

We will be continuing on from the work we did in Lab 2. You don't need to create a new directory or Git repository. Because Git keeps track of all your changes, just continue to work on the same code base. However, you may want to mark this point in your work as the end of lab 2.

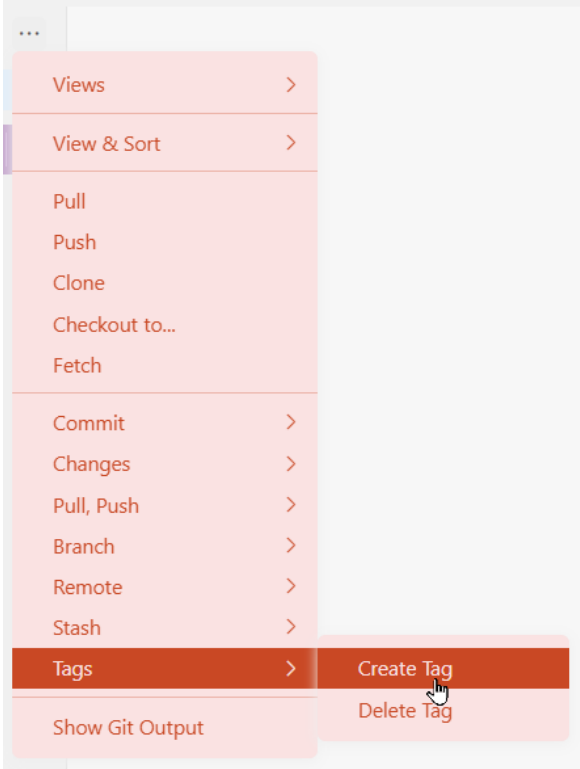
3. In git you can create a tag, which is a checkpoint that marks a point in your project's history. This way you can mark particular versions, e.g. V1, V1.1, V2, etc.

In Visual Studio Code, it's easy to create a tag. Click on the source control icon on the left:

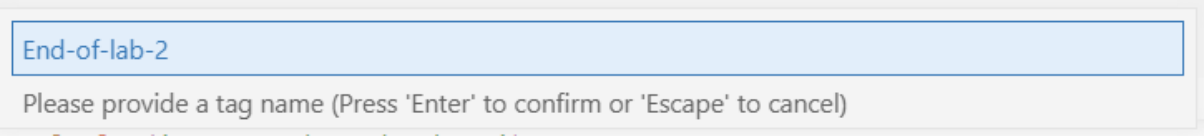
Locate the ... on the source control sub-menu:



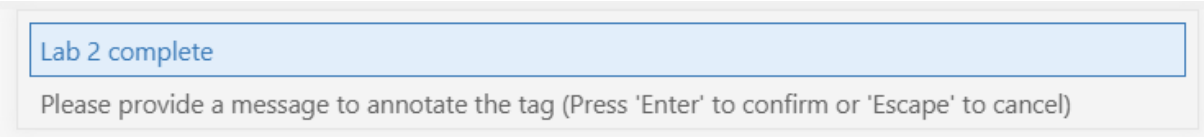
From there you can create a tag:



You can then enter the tag name:



And add an annotation:



4. If you are working from the command line instead of VS Code, you can use git tag:

```
git tag -a End-of-lab-2 -m “Lab 2 complete”
```

5. Git won’t push the tags to the remote unless you explicitly ask it:

```
git push origin --tags
```

Task 3: Install express-session module and import it

6. In order to do session management and authentication install the **express-session** module:

```
npm install express-session
```

For more information related to express-session refer to https://github.com/expressjs/session

7. Import this module in your index.js file:

```
var session = require ('express-session');
```

Task 4: Edit your index.js file

8. Add the following piece of code to your to index.js file to enable sessions. Do you know where to add this code?

```
// Create a session
app.use(session({
    secret: 'somerandomstuff',
    resave: false,
    saveUninitialized: false,
    cookie: {
        expires: 600000
    }
}));
```

What do you think the secret parameter does? Do some reading.

Task 5: Edit your main.js file

9. Add the following piece of code to your main.js file:

```
    const redirectLogin = (req, res, next) => {
        if (!req.session.userId ) {
          res.redirect('./login')
        } else { next (); }
    }
```

Please note, the above piece of code should be added at the beginning of your main.js file after

```
module.exports = function (app, shopData) {
```

so all the routes in main.js can access it.

10. Then add the following lines of code to your 'loggedin' route, where the login is successful:

```
// Save user session here, when login is successful
req.session.userId = req.body.username;
```

11. Now add the 'redirectLogin' to each route that you want to be accessed only by logged-in users as follows:

```
    app.get('/list', redirectLogin, function (req, res) {
```

In this way 'list' page can only be accessed if the user is logged in, otherwise, the user will be redirected to the 'login' page.

You have added access control to your 'list' route, so only logged-in users are able to see the list of the books in your book shop.

12. Run and test your web application:

- First, check the 'list' page without logging in. You should be redirected to the 'login' page and won't be able to see the 'list' page.
- Now do login and then check the 'list' page again, this time you should be able to see the list page.

13. Spend a bit of time reviewing the code you added so that you understand how it works. What is happening behind the scenes here? Think about the HTTP request/response. Use your browser's developer console to inspect the cookies.

Task 6: Add a Logout route to your main.js file

Add the following logout route to your web application.

```
    app.get('/logout', redirectLogin, (req,res) => {
        req.session.destroy(err => {
        if (err) {
          return res.redirect('./')
        }
        res.send('you are now logged out. <a href='+'./'+'>Home</a>');
        })
    })
```

Remember to add the 'logout' 'href' link to your home page as well.

Task 7: Add access control to other pages

14. Add access control to other pages that you think need it.

Run your web application and test and compare accessing all the pages in your web application when logged in and not logged in.

END