T-404-LOKA, Appendix B

Tempo Chrome Extension

Operating Manual

Anton Marinó Stefánsson
Hrönn Róbertsdóttir
Sigrún Þorsteinsdóttir
Þórdís Jóna Jónsdóttir

Spring 2016
Instructor: Haukur Kristinsson
Examiner: Elín Elísabet Torfadóttir

1. Introduction

The Tempo Chrome extension is an extension which simplifies the process of tracking and logging work hours through JIRA. This manual explains the front-end, back-end and testing dependencies used in the project along with how to set up and build the extension, run tests and building a .crx package.

2. Dependencies

Front-end dependencies

React

React is a JavaScript library for building user interfaces. It was made for building large applications with data that changes over time. In React all you do is build encapsulated components, which makes code reuse, testing and separations of concerns easy. React also only reloads the part of the DOM that was modified resulting in better performance and it handles data binding very well.

JSX

JSX is a JavaScript syntax extension that looks similar to XML. It is recommended to use JSX instead of regular JS with React because it is a concise and familiar syntax for defining tree structures with attributes. JSX also gives compile-time errors that, f.ex. tell you when code does not compile due to spelling mistakes.

Redux

Redux uses a unidirectional data flow similar to Flux except it only has a single store. This store is changed by cloning the original store and applying some functions without side effects. This is done without a dispatcher. The stores only source of information is Actions, payloads of information that send data from the application to the store. Actions only describe the fact that something happened, but they do not specify how the application's state changes in response. This is the job of a reducer. The reducer is a pure function that takes the previous state and an action and returns the next state.

Babel

Babel translates ES6/ES7 to the code that runs in the browser, so you can use all of the functions offered by ES6/ES7 even though the browser does not support it yet.

Webpack

Webpack is a module bundler which takes modules with dependencies and emits static assets representing those modules. This is used to reduce the initial loading time of our application. Webpack splits the codebase into multiple chunks and can be loaded at demand.

Back-end dependencies

Node.js

Node.js is an asynchronous event driven framework that is designed to build scalable network applications. Used to proxy requests and keep track of authentication data.

SuperAgent

SuperAgent is a lightweight progressive ajax API that gives flexibility and readability.

3. Tests

Dependencies for tests

Mocha

Mocha is a test framework running on node.js. It has asynchronous testing and it is possible to use any assertion library with it.

4. Development

Setting up the extension

Git clone the project from https://github.com/hronn13/tempoExtension

Navigate to the project folder in your preferred shell and run the command

npm install

next run the command

npm install webpack -g

This will install Webpack globally which is the only dependency in the project which has to be installed globally. The first command will install all other dependencies, which are specified in the file package.json, required to continue development of the project.

Building the extension

Run the command in your preferred shell

webpack --watch

```
hronn@yogi:~/Documents/tempoExtension$ webpack --watch
Hash: 3a9eae317ce43dae8401
Version: webpack 1.12.14
Time: 3534ms
    Asset     Size  Chunks             Chunk Names
bundle.js  1.9 MB       0  [emitted]  app
    + 469 hidden modules
```

This will start a server which runs until it is stopped manually. Each time a file in the project is saved Webpack rebuilds the project.

Go to chrome://extensions

Check the *'Developer mode'* box.

Click the *'Load unpacked extension…'* button.

Locate the project root folder and click the *'Ok'* button.

Make sure the *'Enabled'* box is checked for the extension.

The Tempo logo is now visible near the top of the browser window to the right.

Running tests

Run the command

npm test

All dependencies needed for testing have already been installed with running the command *npm install* (as shown above).

Building a .crx package

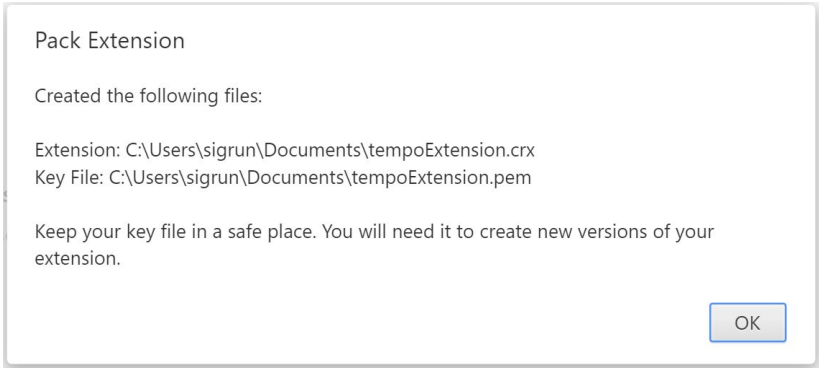
Go to chrome://extensions

Check the *'Developer mode'* box.

Click the *'Pack extension…'* button.

Browse to the project folder that contains the manifest.json file, enter a private key (optional) and click the *'Pack Extension'* button.

The following window will pop up, be patient as this may take a while.



Click the '*OK*' button.

Put the .crx file in the same folder as the manifest.json file.

Uninstall the unpacked version of the Chrome extension so it will not conflict with the packed extension.

Drag and drop the new .crx file into Chrome browser to install the packed app.