Technical Manual

Deploying and Building on Iteration Two

David Wauchope
Devon Rimmington
Faye Teeuwen
Matthew Smith

Twine

Install/Build instructions

Clone the Twine project from the github repository

Install Node.js

- Download the Windows installer from the Nodes.js® web site.
- Run the installer (the .msi file you downloaded in the previous step.)
- Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).
- Restart your computer. You won't be able to run Node.js® until you restart your computer.

(From http://blog.teamtreehouse.com/install-node-js-npm-windows)

Install Twine Dependencies and create a build

- Run "npm install" at the top level of the directory to install all goodies.

Building Twine

- Enter the terminal (Linux, OSX) or run cmd (windows)
- Enter the directory bioethics-web-app/twine
- Run check the directory and make sure the file grunt is present
- Run the command "Grunt"
- A fully built project can be found under twine/build/…
    - If the javascript doesn't compile then a descriptive error message will be shown and the build will fail.

Vue Framework

Vue is a web framework for making interactive web applications. It's very simple to use, below are some basics that should be reviewed before continuing in this document.

Vue documentation https://vuejs.org/v2/guide/ Under "Essentials" on the left side review:

- Introduction
- Template Syntax
- Computer Properties and Watchers
- Conditional Rendering
- Event Handling
- Components

Within the project under the **Twine** directory the most important subdirectories for getting started are:

- Src: this is where all of the source code goes; in here you can find html and javascript files that the project runs off of.
- Test: this is where all of the unit tests are; it's a good place to look to trace a function back to its source.
- Build: this is where the html project will build to when you run the grunt command.
- Story-format: contains minified js files; these contain the code for presenting the story to the user.

Under the src directory are subdirectories that will typically either contain only javascript files (these are where code functionality is stored such as save and load files) or index files both html/javascript.

If it contains index files you will see an index.html and index.js files and maybe more subdirectories that contain more of the same.

The lower level html files are templates for rendering html elements to the screen and the necessary javascript code to run it.

An example of this structure is src/story-edit-view (yellow means directory)

- Story-edit-view

- Index.html
- Index.js
- Index.less
- Zoomsettings.js
- Story-tool-bar
  - Index.html
  - Index.js
  - Index.less
  - Story-menu
  - Story-search

The uppermost index.html uses its accompanying index.js to set/create custom html elements using the lower html files as the template.

An important file that we have made some modifications to **bioethics-web-app\twine\src\file\save.js on line 42** we have added an ajax call that will send the title (.html will need to be stripped from the title string) and the html data itself to a webservice or script. This might need to be changed when adding authentication, however, as a starting off point it will work.

Recommended Next Steps

Overview

For future development, our project will require a database implementation to store some key data. Our database schematic is shown in figure 1. This database stores all necessary user information, as well as relevant health case information for Twine to use. Using the user information, an authentication system will need to be implemented to allow content submission and editing. We do not recommend you build one from scratch, you should consider pre-existing PHP authentication system options before designing an authentication system. A front end user interface will also need to be designed. Included in our project is a rudimentary bootstrap template that can be implemented for the start of the user interface. Finally, Twine will need to be integrated smoothly into the database and the front end before finishing touches and additional features can be added.

Website Development Recommendations

During the second iteration we attempted to integrate Twine with the Django framework, however, we learned that the systems were fundamentally incompatible in relation to their routing systems for getting static Javascript files. Hence, it is recommended that the next group uses a more customizable solution such as plain PHP or perhaps the PHP framework Laravel to build a website. One benefit of using a framework like Laravel is that it removes the need to build an authorization system, however any chosen framework must be capable of integration with Twine.

Current Server Requirements as Created by Phase 1

The first iteration group had a clear vision of the project using Django, unfortunately it did not meet the client's needs of creating interactive cases. As a result they have a server setup with the following criteria:

- *The project needs an Apache server and mod_wsgi*
- *It needs Python which also handles a database, sqlite*
- *It's likely these are set up on a Linux or CentOS server at Dalhousie*
- *The student team can setup any requirements within the space if needed*
- *The project is built using Django, a Python framework*
- *The majority of the content will be text based, but may include some images*

It is recommended that the iteration three group put in a request to Marika to have the server customized for their needs, as ascertained by their choice of technology for iteration three. SQLite is a very reasonable database system to use for medium load (the server won't have even close to that kind of load).

Tables and Figures

Figure 1