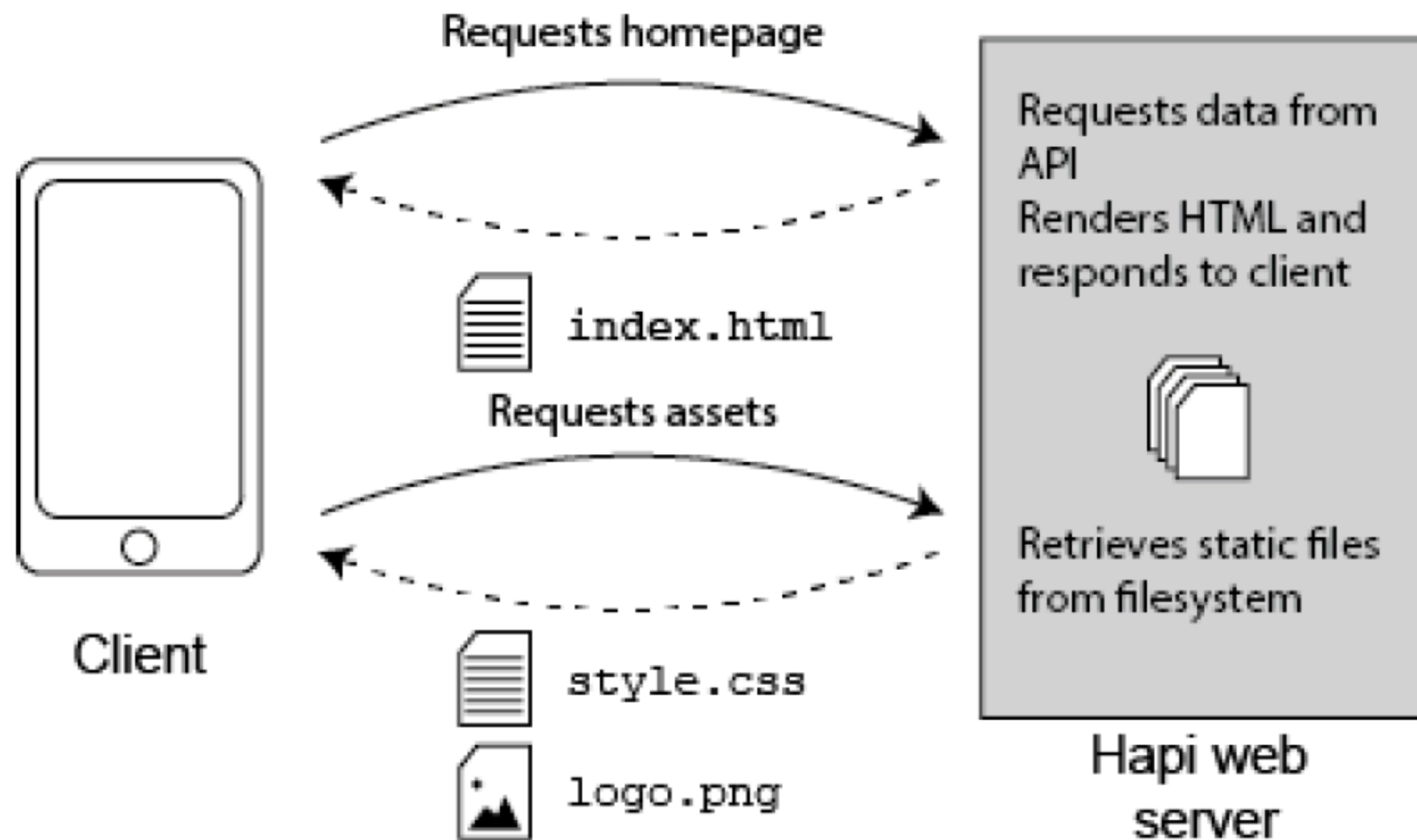


Views

Application Structure

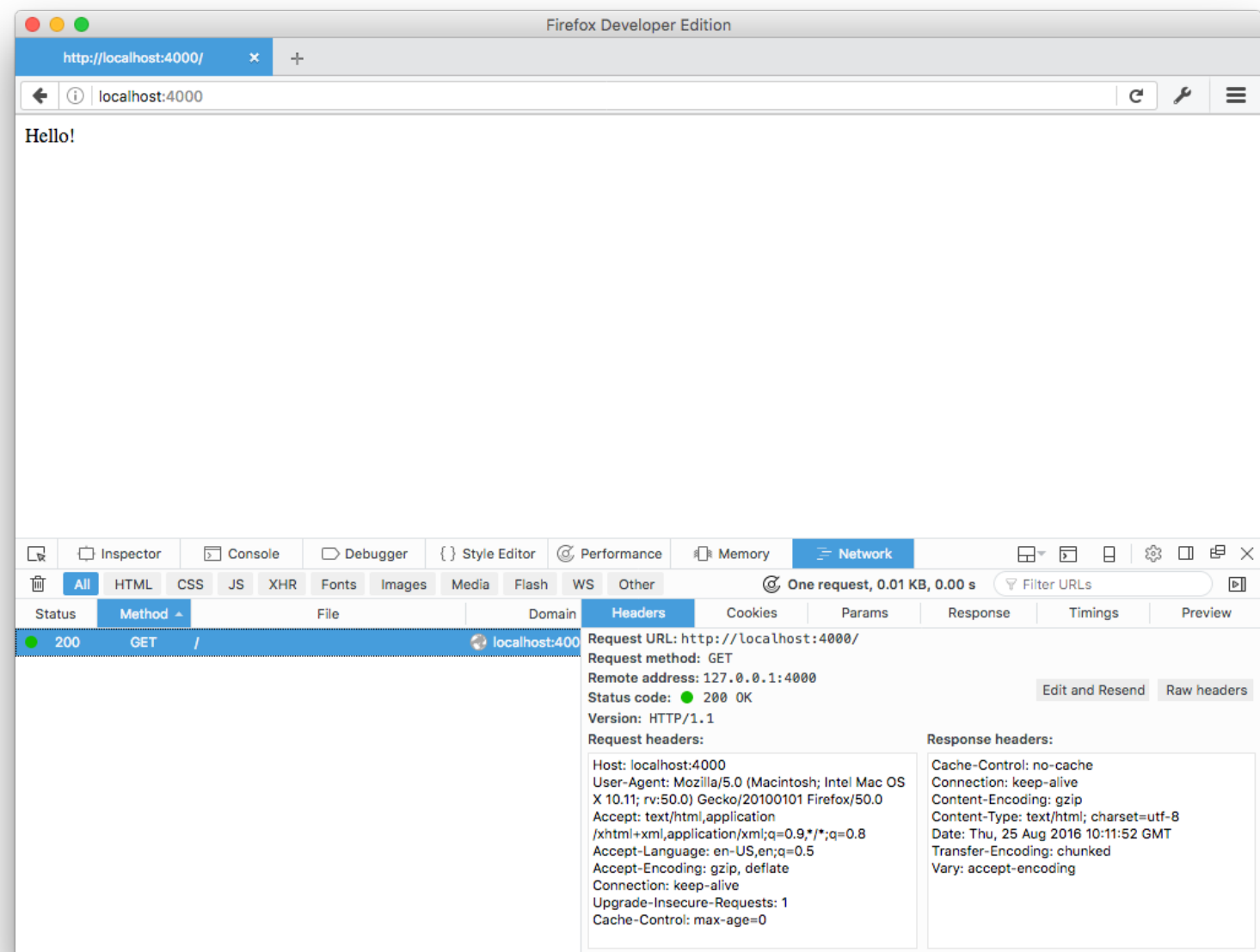


reply

- reply responds to the browser with a simple string.

controller.js

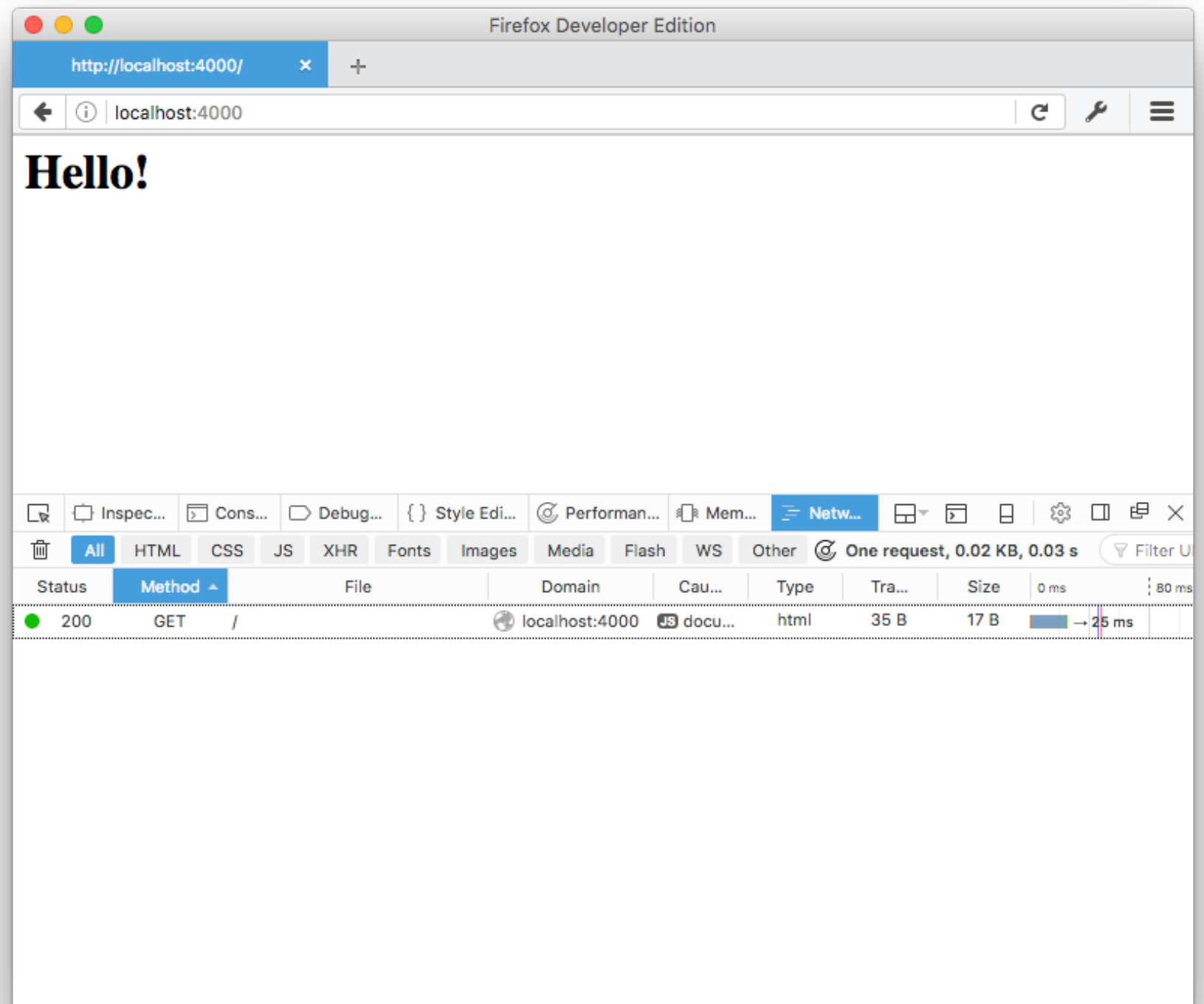
```
exports.index = {  
  
  handler: function (request, reply) {  
    reply('Hello!');  
  }  
  
};
```



reply

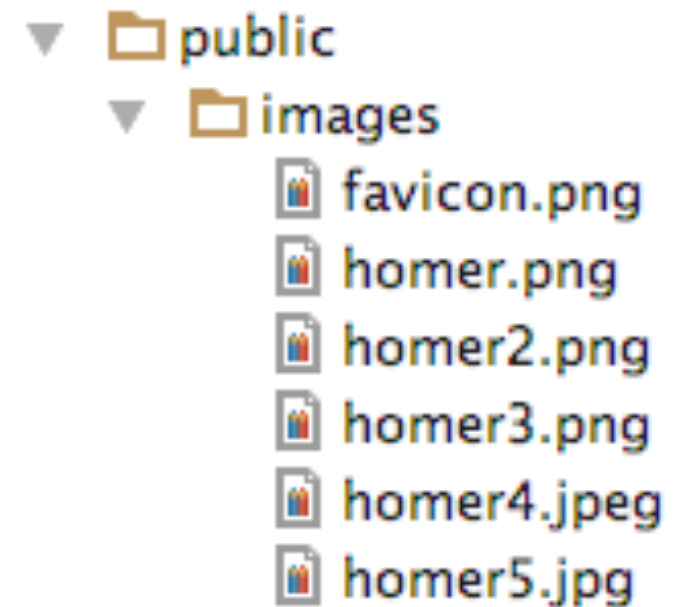
- In order to render web pages we could pass html content
- This would become very unwieldy and unmaintainable

```
exports.index = {  
  handler: function (request, reply) {  
    reply('<h1> Hello! </h1>');  
  }  
};
```



Static Assets

- Additionally, serving folders of static content (images, css etc..) might require an extensive range of routes + handlers



inert plugin: <https://github.com/hapijs/inert>

- This plugin provides simple file serving features for hapi

inert

Static file and directory handlers plugin for hapi.js.

build passing

Lead Maintainer - [Gil Pedersen](#)

inert provides new [handler](#) methods for serving static files and directories, as well as decorating the [reply](#) interface with a `file` method for serving file based resources.

Features

- Files are served with cache friendly `last-modified` and `etag` headers.
- Generated file listings and custom indexes.
- Precompressed file support for `content-encoding: gzip` responses.
- File attachment support using `content-disposition` header.

Index

- [Examples](#)
 - [Static file server](#)
 - [Serving a single file](#)
 - [Customized file response](#)
- [Usage](#)
 - [Registration options](#)
 - `reply.file(path, [options])`
 - The `file` handler
 - The `directory` handler

<https://github.com/hapijs/inert>

Restructure the Application Project Structure

- As we are about to grow a more substantial projects, we take care to lay out a sustainable folder structure.

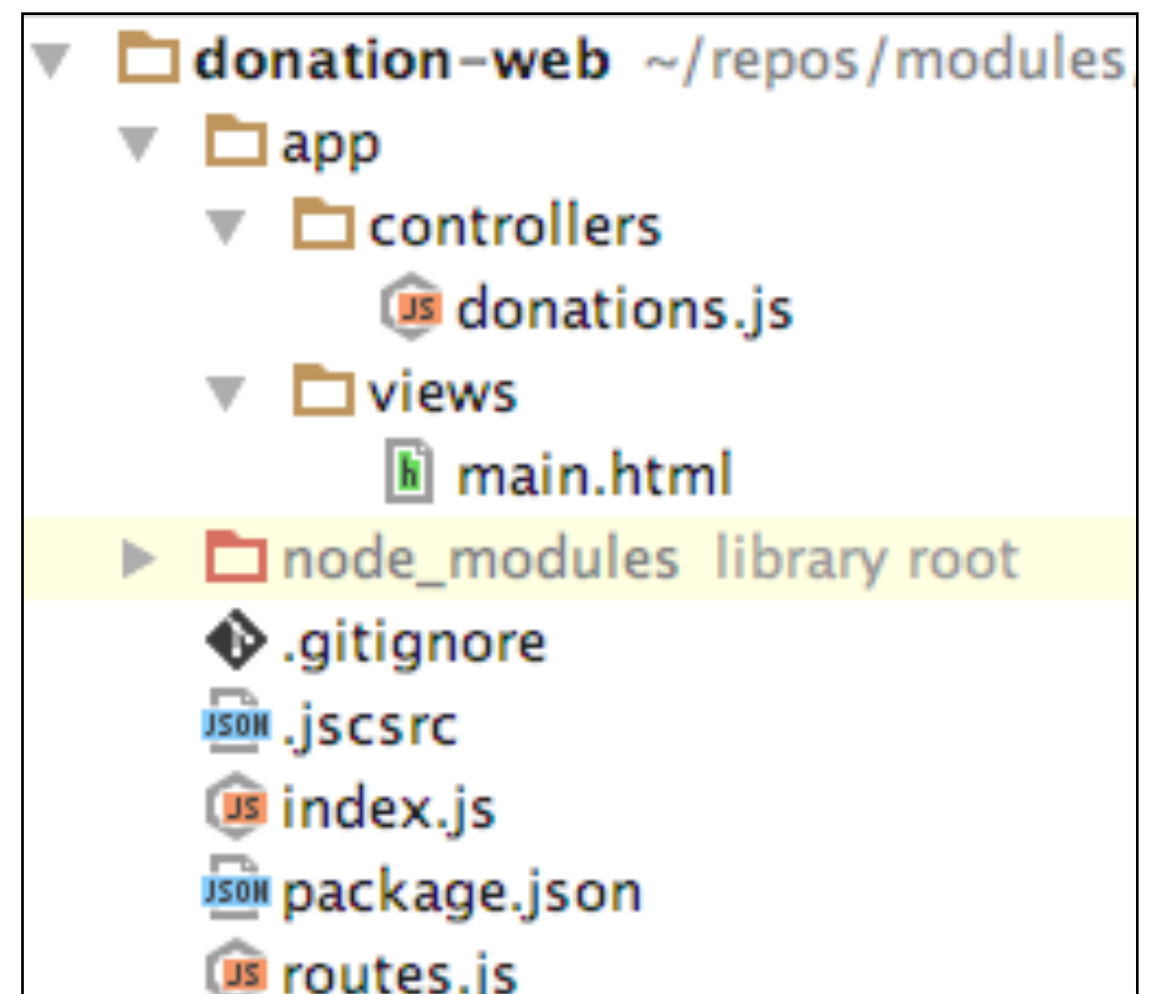
app contents bulk of application sources

controllers will contain handlers

views will contain html templates

index.js is the application entry point

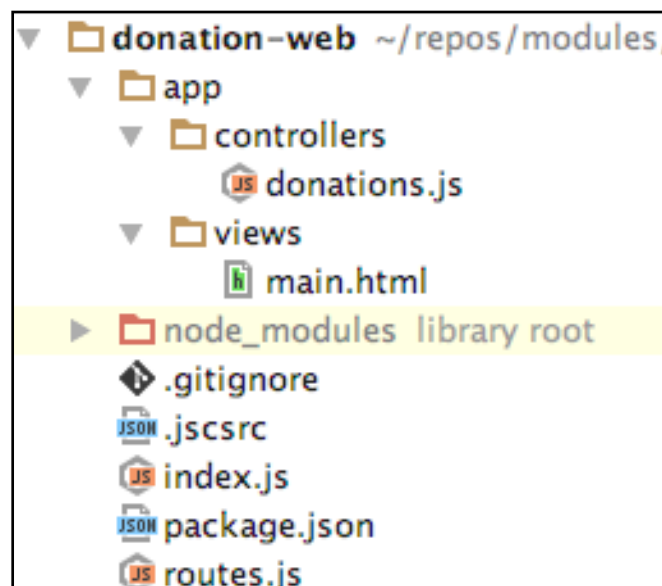
routes.js define all application routes



Main Application Entrypoint

index.js

- Plugin must be registered, then if successful:
 - Routes included
 - Server started



```
'use strict';

const Hapi = require('hapi');

var server = new Hapi.Server();
server.connection({ port: process.env.PORT || 4000 });

server.register(require('inert'), err => {

  if (err) {
    throw err;
  }

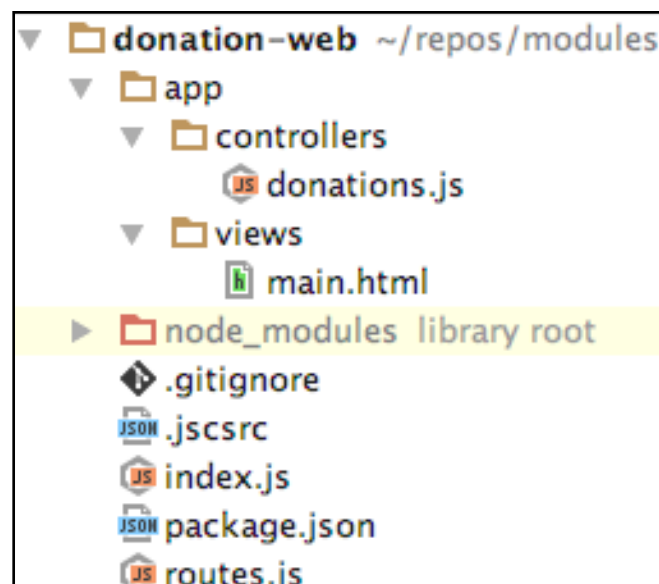
  server.route(require('./routes'));
  server.start((err) => {
    if (err) {
      throw err;
    }

    console.log('Server listening at:', server.info.uri);
  });
});
```

\$ npm install inert -save

Routes + Controller

- Route recognises a single pattern: '/'.
- The handler replies with a single file - which will be rendered in the client browser



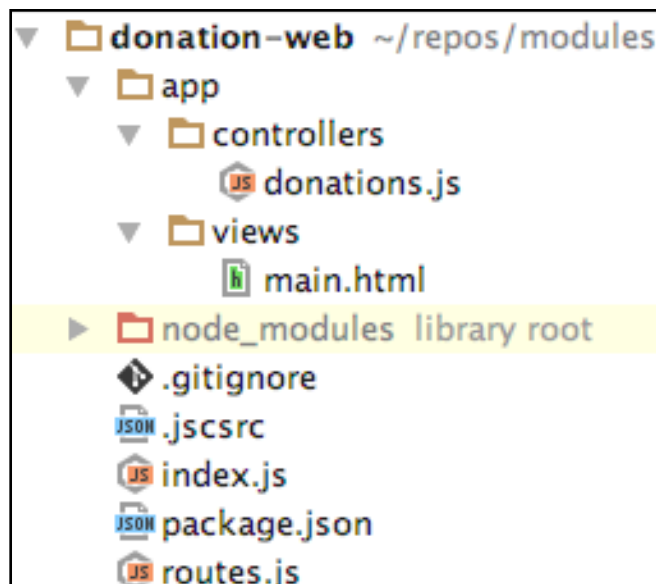
routes.js

```
const Donations = require('./app/controllers/donations');  
  
module.exports = [  
  { method: 'GET', path: '/', config: Donations.home },  
];
```

app/controllers/donation.js

```
'use strict';  
  
exports.home = {  
  handler: (request, reply) => {  
    reply.file('./app/views/main.html');  
  },  
};
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Donations</title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.1.6/semantic.min.js"></script>
    <link rel="stylesheet" media="screen" href="//cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.1.6/semantic.min.css">
  </head>
  <body>
    <section class="ui container">
      <section class="ui stacked segment">
        <div class="ui grid">
          <aside class="six wide column">
            
          </aside>
          <article class="ten wide column">
            <header class="ui header"> Help Me Run Springfield</header>
            <p> Donate what you can now – No Bitcoins accepted! </p>
          </article>
        </div>
      </section>
    </section>
  </body>
</html>
```



- Simple, static web page.
- Loads Semantic UI Stylesheet from CDN.

Semantic UI

- Rich, themeable, attractive layouts & UI components

Unbelievable Breadth

Definitions aren't limited to just buttons on a page. Semantic's components allow several distinct types of definitions: elements, collections, views, modules and behaviors which cover the gamut of interface design.

[See Layout Examples >](#)

Menu

Menu component example showing a list of items (Inbox, Trash, Search mail...) with counts (1, 1) and a search bar.

Card

Card component example showing a user profile card for Rachel Maddaw, Pundit, with 22 Friends and joined in 1998.

Dropdown

Dropdown component example showing a country selection dropdown and a filter posts button.

Segment

Segment component example showing a horizontal list of segments.

Divider

Divider component example showing a horizontal line with text 'A OR B'.

Feed

Feed component example showing a list of social media posts (Mark added you as a friend, You added Lena to the group Close Friends, Eve just posted on your page).

Input

Input component example showing a search bar and a URL input field.

Accordion

Accordion component example showing a list of items (Size, Colors) with expand/collapse functionality.

Label

Label component example showing a user profile card with a label (molly@thebears.com).

Checkbox

Checkbox component example showing a list of checkboxes (I enjoy having fun, Receive weekly poodle newsletter, Make my dog's profile public).

Message

Message component example showing a cookie notice and a help link.

Step

Step component example showing a multi-step form for shipping, billing, and confirming order.

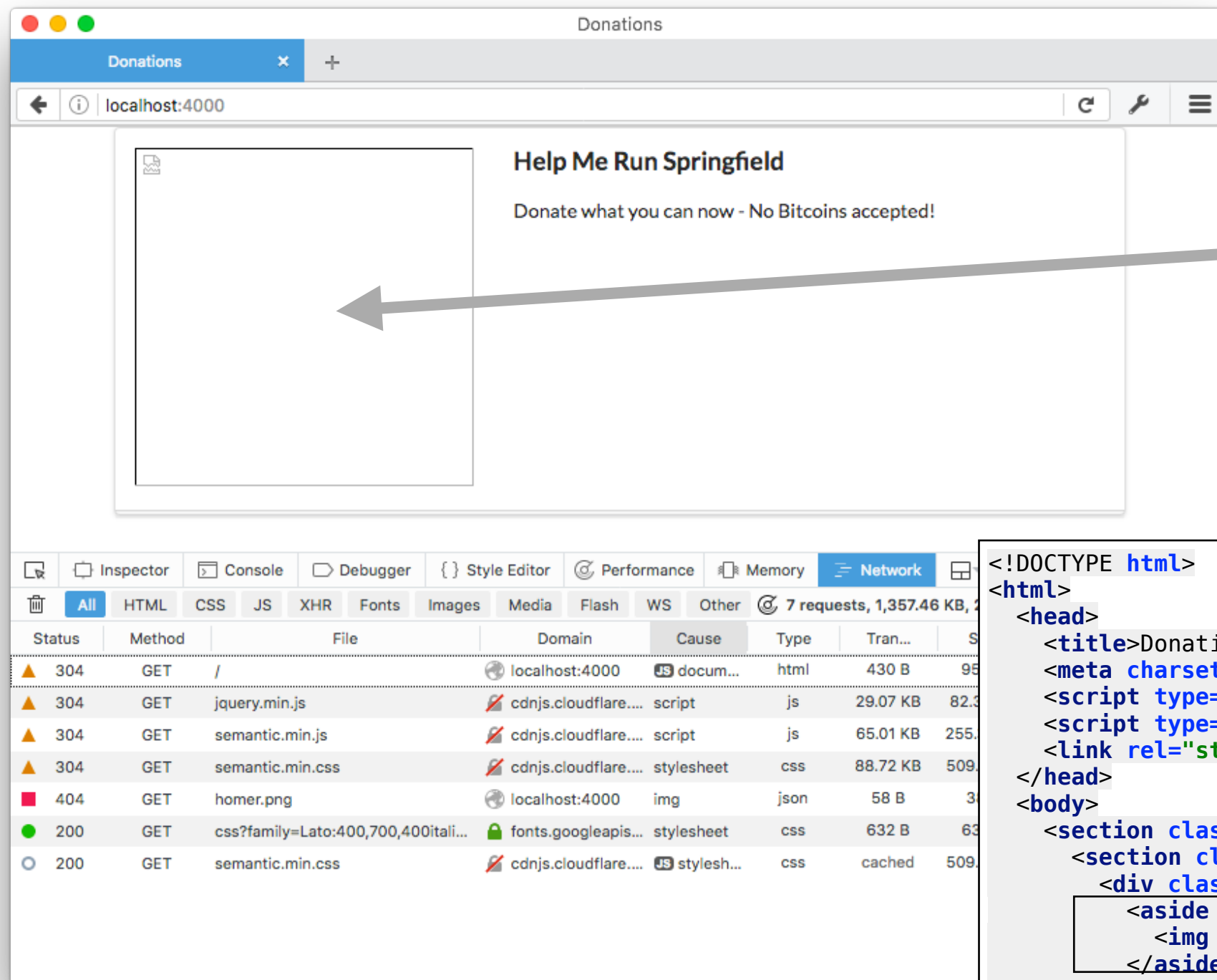
Examples to introduce components and their function

Examples to introduce components and their function: Theming, Responsive, Grid, Attached Content, Bootstrap Migration.

Pages

Starter page templates

Starter page templates: Homepage, Sticky Menus, Fixed Menu, Login Form.



- Static resource not served

```
<!DOCTYPE html>
<html>
  <head>
    <title>Donations</title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.2.8/semantic.min.js"></script>
    <link rel="stylesheet" media="screen" href="//cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.2.8/semantic.min.css">
  </head>
  <body>
    <section class="ui container">
      <section class="ui stacked segment">
        <div class="ui grid">
          <aside class="six wide column">
            
          </aside>
          <article class="ten wide column">
            <header class="ui header"> Help Me Run Springfield</header>
            <p> Donate what you can now – No Bitcoins accepted! </p>
          </article>
        </div>
      </section>
    </section>
  </body>
</html>
```

routes.js

```
const Donations = require('./app/controllers/donations');
const Assets = require('./app/controllers/assets');

module.exports = [

  { method: 'GET', path: '/', config: Donations.home },

  {
    method: 'GET',
    path: '/{param*}',
    config: { auth: false },
    handler: Assets.servePublicDirectory,
  },

];
```

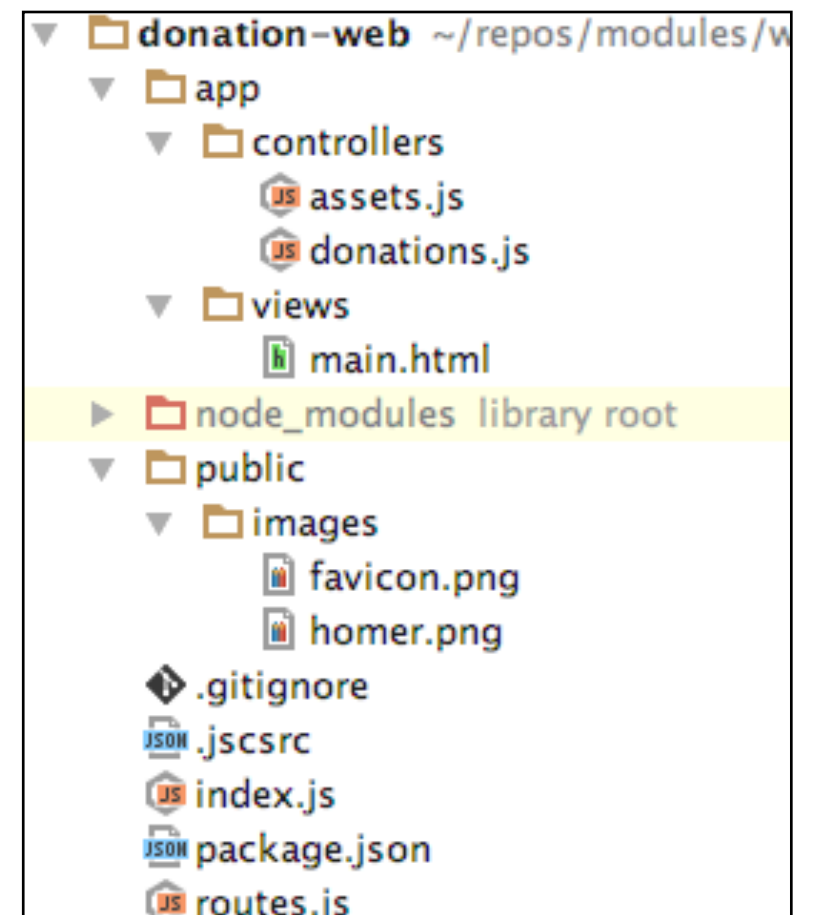
Static Assets

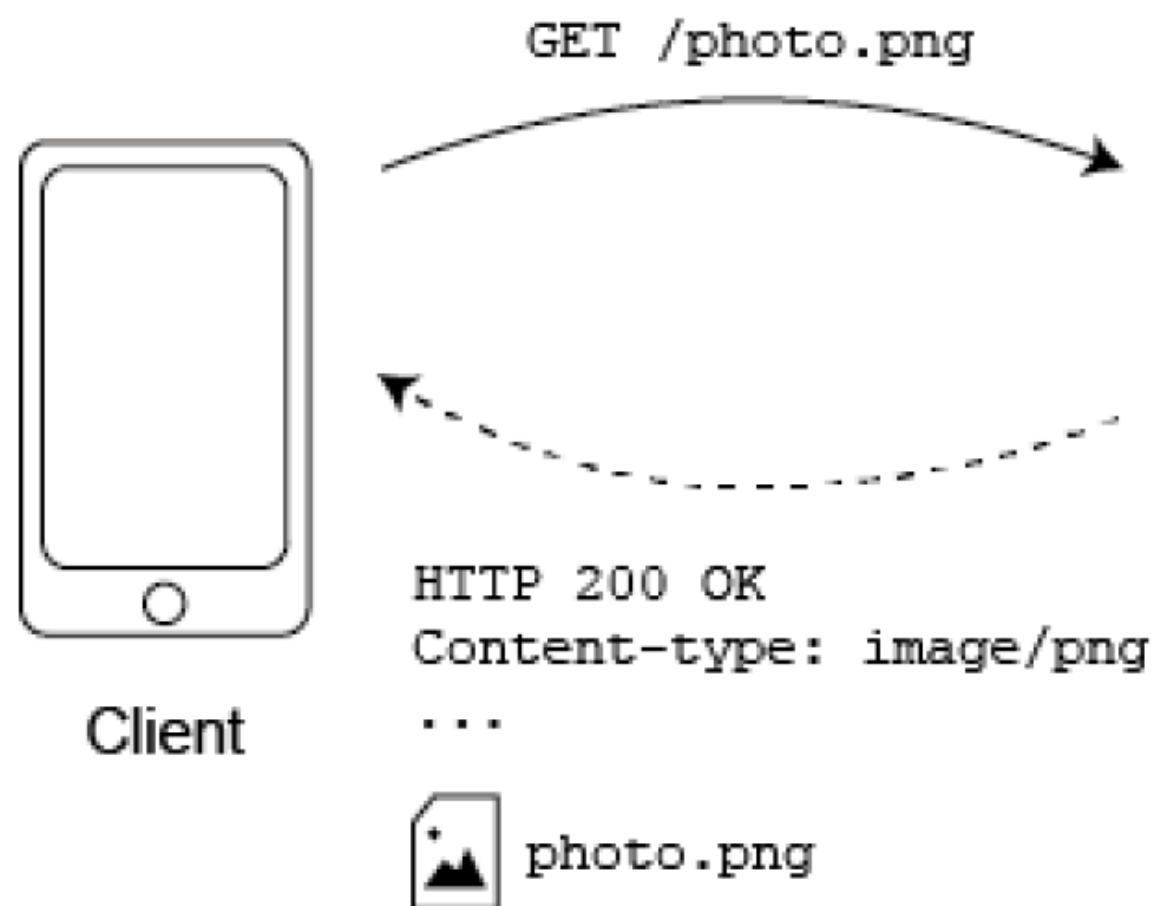
- additional route + handler

app/controllers/assets.js

```
'use strict';

exports.servePublicDirectory = {
  directory: {
    path: 'public',
  },
};
```





Route

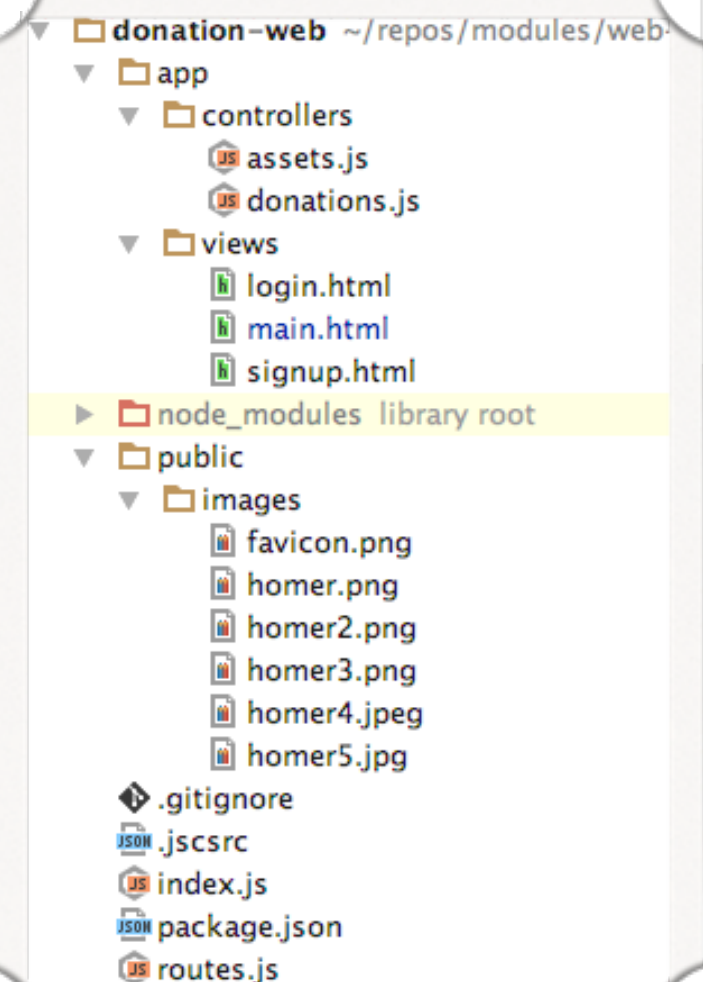
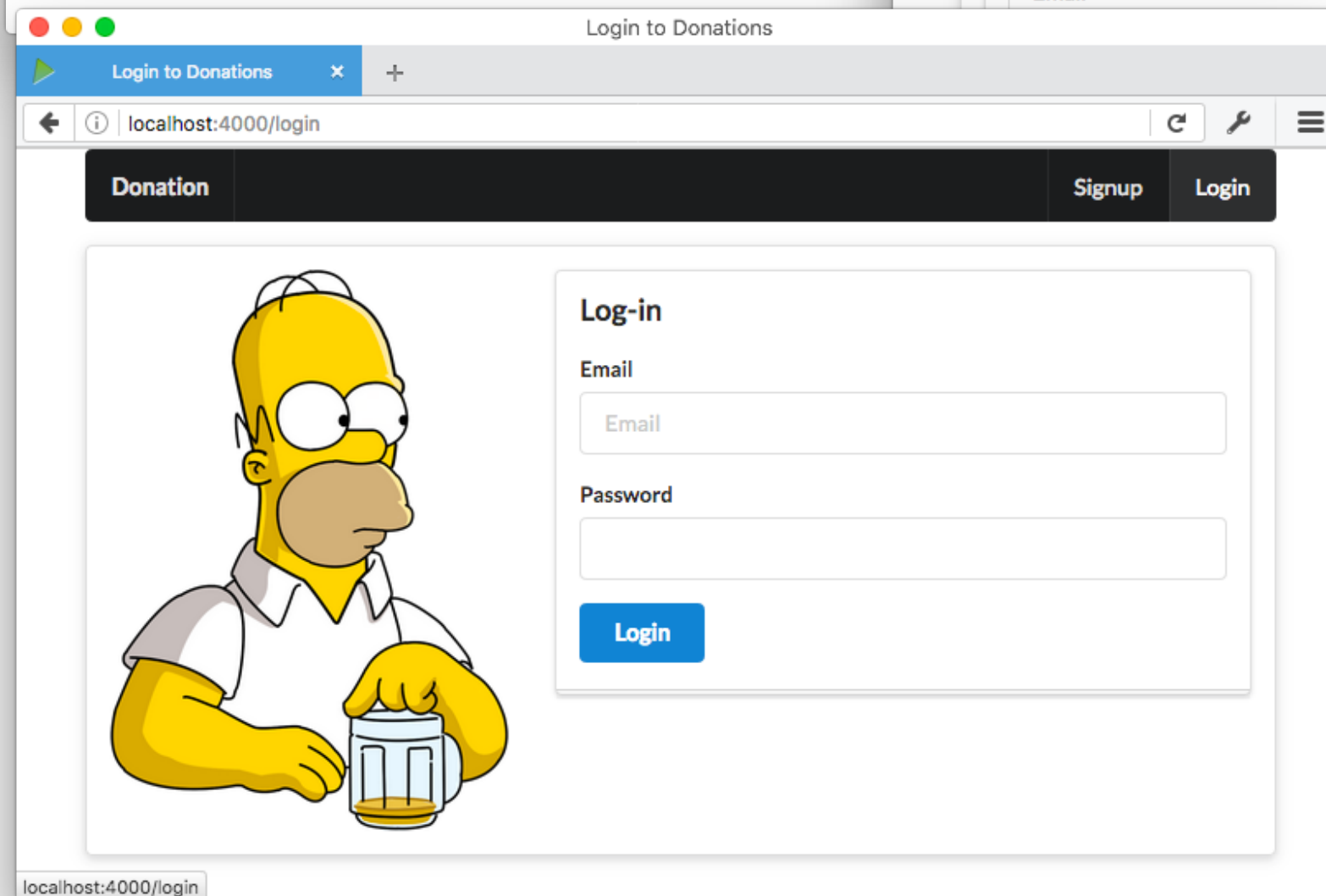
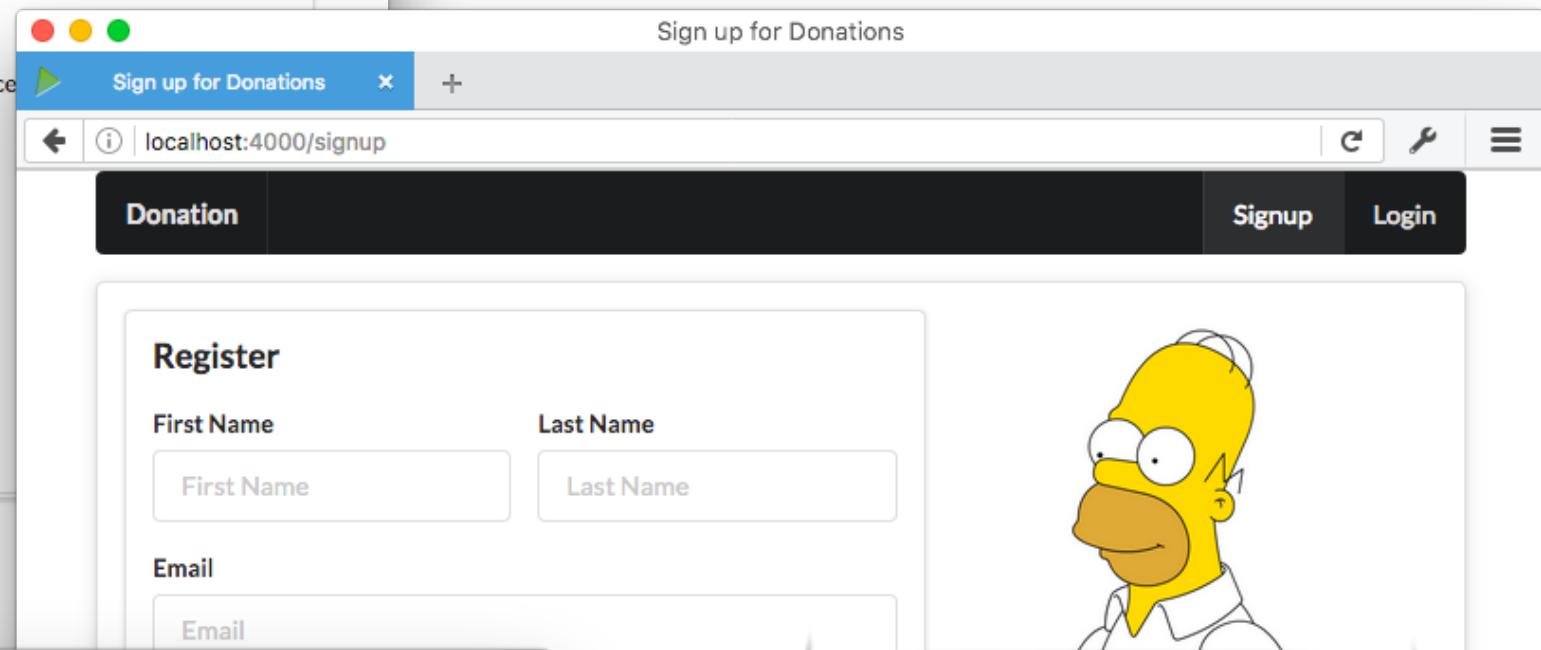
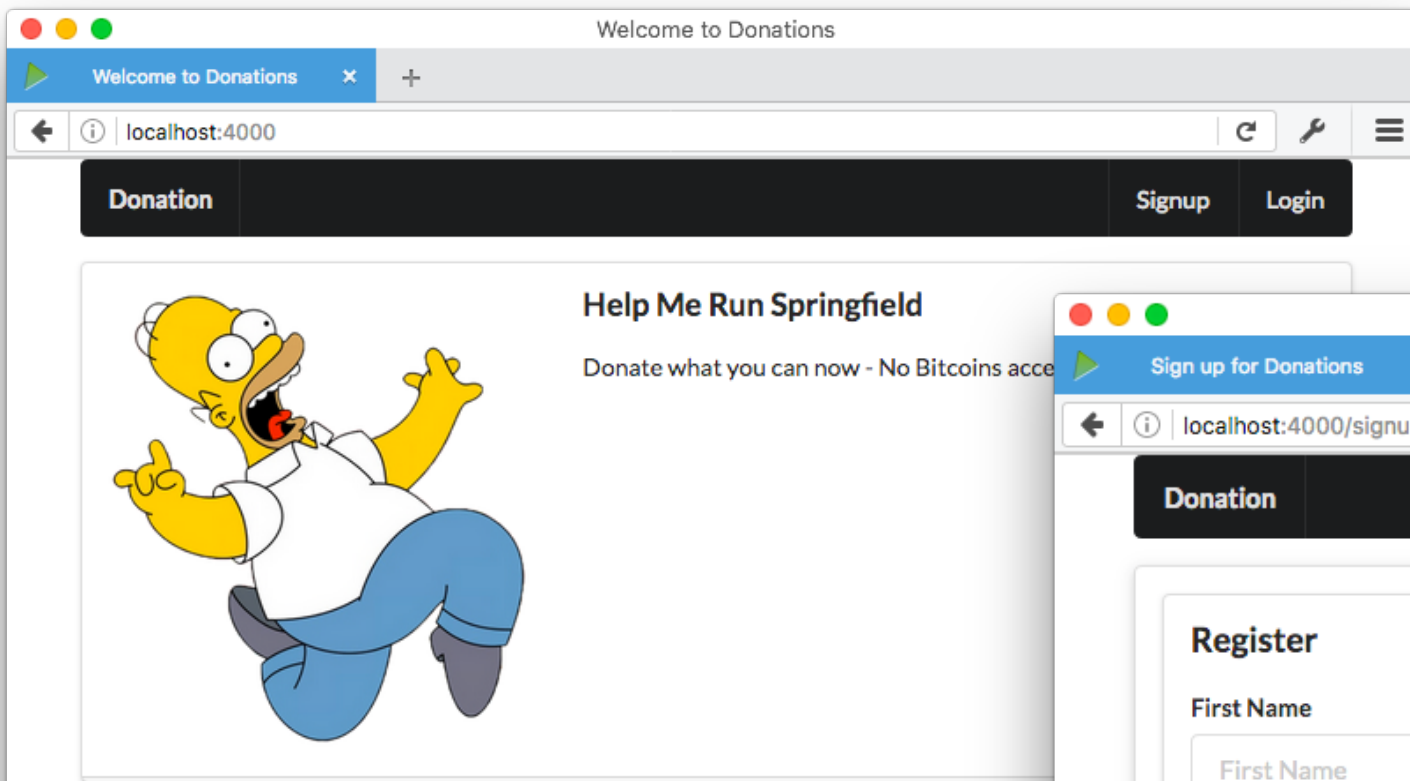
```
server.route({  
  method: 'GET',  
  path: '/photo.png',  
  handler: function (request, reply) {  
    ...  
    reply.file('photo.png');  
  }  
});
```



Hapi looks in filesystem for a file named photo.png and responds with the file's contents

Hapi web server

Skeleton App UI



routes.js

```
const Donations = require('./app/controllers/donations');
const Assets = require('./app/controllers/assets');

module.exports = [

  { method: 'GET', path: '/', config: Donations.home },
  { method: 'GET', path: '/signup', config: Donations.signup },
  { method: 'GET', path: '/login', config: Donations.login },

  {
    method: 'GET',
    path: '/{param*}',
    config: { auth: false },
    handler: Assets.servePublicDirectory,
  },
];
```

app/controllers/donations.js

```
'use strict';

exports.home = {

  handler: (request, reply) => {
    reply.file('./app/views/main.html');
  },
};

exports.signup = {

  handler: (request, reply) => {
    reply.file('./app/views/signup.html');
  },
};

exports.login = {

  handler: (request, reply) => {
    reply.file('./app/views/login.html');
  },
};
```

Project Structure

