puerF

Commandline interface to use puer https://github.com/leeluolee/puer with FreeMarker templates http://freemarker.org/.

Usage

Install it either as a global command line tool or as a local development dependency:

```
npm install -g puer-freemarker
or
npm install --save-dev puer-freemarker
```

Move into your working directory and run it:

```
cd your/working/directory
puerf
```

puerF requires that you have Java installed as it is needed to render FreeMarker templates.

Command reference

```
Usage: puerf [cmd] [options]

Commands:

   init [options]   Set up basic folders and files to work with puerf

Start a puer Server, easily mock routes and render FreeMarker templates

Options:

   -h, --help              output usage information
   -V, --version           output the version number
   -r, --routes <file>     Configuration file for mocked routes (multiple possible)
   -c, --config            If a config file should be used
   -t, --templates <path>  Path to folder in which Freemarker templates are stored
   -r, --root <folder>     The root folder that files should be served from
   -p, --port <number>     Specific port to use
   -w, --watch <files>     Filetypes to watch, defaults to js|css|html|xhtml|ftl
   -x, --exclude <files>   Exclude files from being watched for updates
   -l, --localhost         Use "localhost" instead of "127.0.0.1"
   --no-browser            Do not autimatically open a brwoser
   --debug                 Display debug messages
```

Mocking requests

This is what puerF is really all about. Making it as easy as possible for you to "fake" a backend. To achieve this puerF builds upon puers mocking of request https://github.com/leeluolee/puer#mock-request. And simplifies the use of FreeMarker templates for those requests.

To mock requests create modules that export a single object following the syntax outlined below. You may split your routes over as

many files as and modules as you which, puerF will combine them for you. `demos` contains a sample file.

By default puerF will assume that you spread your routes over two files `mock/routes.js` and `mock/ftlRoutes.js`.Should you wish to use files from a different location you can do so using the `-r` option.

Syntax for routes

All routes should be described as a String identifying the route and a `routeObject` containing information about how to mock that route. The `routeObject` may have any of the following properties:

Property	Description
handler	A function that handles this route
template	A FreeMarker template to be rendered for this route
data	An object to be returned at this route or used for the template
jsonFile	The path to a file containing the `data` for this route

When a route is called puerF resolves the `routeObject` as such:

- If a handler is present, it is executed
- Else if a template is specified it is parsed
- If none of the above are true the data is returned

Working with query parameters

Real world applications might use query parameters to get specific results from a URL. You can easily mock those requests as well. To mock a route like `/example/some?user=name`, inside your regular routes file, you can simply access `req.query.user` to get the users name.

```
"GET /example/some": {
    handler: function(req, res, next) {
        var name = req.query.user;

        //Do something with the name, like sending it back.
        res.status(200).send(name).end();
    }
}
```

FreeMarker templates

By default puerF will look for FreeMarker templates in `./templates`. To specify another folder you can use `-t`. Read the guid to author FreeMarker templates http://freemarker.org/docs/dgui.html.

Use as a dependency

Instead of using `puer-freemarker` as a commandline tool, you might also use it as a dependency in your development pipeline. For instance in your gulpfile.

To do so simply `var puerf = require('puer-freemarker')`. The full docs may be found here http://hoverbaum.github.io/puerF. Below is the public API of puerF.

puer-freemarker

puerF, a simple tool to run a live reloading server with mocked routes and FreeMarker templates.

- puer-freemarker
  - .init(options, callback)
  - .start(options, callback)
  - .close(callback)

puer-freemarker.init(options, callback)

Runs the initializer. Will output basic files to the current working directory.

Kind: static method of `puer-freemarker`
See: initializer

Param	Type	Default	Description
options	`object`		Options for the initializer.
[options.onlyConfig]	`boolean`	`false`	Only create the config file.
callback	`function`		Function to call once done.

puer-freemarker.start(options, callback)

Starts the core application.

Kind: static method of `puer-freemarker`

Param	Type	Default	Description
options	`object`		An object containing options.
[options.routes]	`array`	`['mock/ftlRoutes.js', 'mock/routes.js']`	An array of paths to all files containing mocked routes.
[options.config]	`boolean`	`false`	Use config file.
[options.templates]	`string`	`"'templates'"`	Root folder for FTL template files.
[options.root]	`string`	`"'./'"`	Root folder for static files to serve.
[options.port]	`number`	`8080`	The port to use for the server.
[options.watch]	`string`	`"'js css html xhtml ftl'"`	Filetypes (seperate them by pipes) to watch for changes.
[options.exclude]	`regEx`	`/node_modules/`	Files to exclude from watching.
[options.localhost]	`boolean`	`false`	Use `localhost` instead of `127.0.0.1`.
[options.browser]	`boolean`	`true`	Automatically open a browser for the user.

[options.debug]	`boolean`	`false`	Enable debugging output and log file.
callback	`function`		Function to call once started.

puer-freemarker.close(callback)

Programatically closes puerF.

Kind: static method of `puer-freemarker`

Param	Type	Description
callback	`function`	Function to call once done.

Testing

puerF utilizes tape https://github.com/substack/tape for testing. Install development dependencies and run `npm test` to run the tests.