mean-stack (0.0.1)

Maksim Kostromin

Version 0.0.1, 2018-06-25 05:57:50 UTC

Table of Contents

Travis CI status:

Chapter 1. Introduction

Read reference documentation

- Initial setup: Angular 6 (frontend)
- TODO: Express + MongoDB (backend)

Chapter 2. Implementation

2.1. frontend

add material design to project:

```
ng add @angular/material
```

configure material design as separed module (`app-angular-material.module.ts` file):

```
// import { NoopAnimationsModule } from '@angular/platform-browser/animations';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { MatToolbarModule } from '@angular/material';

const modules = [
  // NoopAnimationsModule,
  BrowserAnimationsModule,
  MatToolbarModule,
];

@NgModule({
  imports: modules,
  exports: modules,
})
export class AppAngularMaterialModule { }
```

configure application routes (`app-routing.module.ts` file):

```
import { Routes, RouterModule } from '@angular/router';
import { CreateComponent } from './components/create/create.component';
import { EditComponent } from './components/edit/edit.component';
import { ListComponent } from './components/list/list.component';

const routes: Routes = [
  { path: 'create', component: CreateComponent },
  { path: 'edit/:id', component: EditComponent },
  { path: 'list', component: ListComponent },
  { path: '**', redirectTo: 'list', pathMatch: 'full' },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Chapter 3. Docker

3.1. frontend

build frontend application

```
npm i
npm run dist
```

build docker image based on nginx

```
cp -Rf ./.dockerignore.frontend ./.dockerignore
docker build -f Dockerfile.frontend -t daggerok/mean-stack:frontend ./frontend
```

run application in docker

```
docker run -d --rm -p 80:80 -e "PORT=80" --name run-mean-stack-frontend daggerok/mean-stack:frontend
```

check application is working on 0.0.0.0/

```
open http://127.0.0.0/
```

finally stop docker container for cleanning up...

```
docker stop run-mean-stack-frontend
```

Chapter 4. Heroku deployment

4.1. frontend

4.1.1. prepare nginx in docker for heroku

Idea is very simple: as far nginx doesn't support environment variables, but heroku is required ENV `$PORT` to be used, we can create nginx default.conf template and during build or bootstrap generate and replace it in nginx. see `Dockerfile.frontend` for derails

We will be using `Dockerfile.frontend-build` file instead. This file also contains instructions for project build using npm.

prepare frontend Dockerfile (`Dockerfile.frontend` file):

```
FROM nginx:1.15.0-alpine
LABEL MAINTAINER="Maksim Kostromin https://github.com/daggerok"
ARG PORT_ARG="80"
ARG HOST_ARG="localhost"
ENV PORT=${PORT_ARG}
ENV HOST=${HOST_ARG}
RUN apk add --update --no-cache libintl curl \
 && apk add --no-cache --virtual build_deps gettext \
 && cp /usr/bin/envsubst /usr/local/bin/envsubst \
 && apk del build_deps \
 && echo '#!/bin/sh' >> /usr/local/bin/entrypoint.sh \
 && echo 'PORT=${PORT:=80}' >> /usr/local/bin/entrypoint.sh \
 && echo 'HOST=${HOST:=localhost}' >> /usr/local/bin/entrypoint.sh \
 && echo '#/bin/sh -c "envsubst < /root/default.conf.tpl >
/etc/nginx/conf.d/default.conf && exec nginx -g \"daemon off;\""' >>
/usr/local/bin/entrypoint.sh \
 && echo '/bin/sh -c "envsubst < /root/default.conf.tpl >
/etc/nginx/conf.d/default.conf && (cat /etc/nginx/conf.d/default.conf | grep listen)
&& (cat /etc/nginx/conf.d/default.conf | grep server_name) && exec nginx -g \"daemon
off;\""' >> /usr/local/bin/entrypoint.sh \
 && chmod +x /usr/local/bin/entrypoint.sh
ENTRYPOINT entrypoint.sh
CMD /bin/bash
COPY ./docker/default.conf.tpl /root/default.conf.tpl
COPY ./frontend/dist/frontend /usr/share/nginx/html
```

prepare nginx config template (`docke/default.conf.tpl` file) to be used by Dockerfil during image build:

```
server {
  listen       ${PORT};
  server_name ${HOST};
  charset     utf-8;

  location / {
    root  /usr/share/nginx/html;
    index index.html index.htm;
    allow all;
  }

  # redirect some server error pages to the fallback page /index.html
  error_page 400 404 500 502 503 504 /index.html;
  location = /index.html {
    root  /usr/share/nginx/html;
    allow all;
  }
}
```

4.1.2. prepare heroku config

prepare `heroku.yml` file:

```
build:
  docker:
    web: Dockerfile.frontend
```

4.1.3. create and deploy heroku app

build app

```
cd ./frontend
npm i
npm run dist
cd ..
```

create heroku application

```
heroku create daggerok-mean-stack-frontend --stack=container

# output:

Creating    daggerok-mean-stack-frontend... done, stack is container
https://daggerok-mean-stack-frontend.herokuapp.com/ | https://git.heroku.com/daggerok-
mean-stack-frontend.git
```

push app

```
cp -Rf ./.gitignore.heroku ./.gitignore
cp -Rf ./heroku.yml.frontend ./heroku.yml
cp -Rf ./.dockerignore.frontend ./.dockerignore

git add .
git commit -am "Deploy heroku app."
git push heroku master
```

Chapter 5. docker can also build...

Here we will be using `Dockerfile.frontend-build` file instead. This file also contains instructions for project build using npm. We wanna simply delegate everything to `git push heroku master` instruction

prepare frontend-build Dockerfile (`Dockerfile.frontend-build` file):

```
FROM node:8.11-alpine
LABEL MAINTAINER="Maksim Kostromin https://github.com/daggerok"
WORKDIR /root/mean-stack/frontend
COPY ./frontend .
COPY ./docker/default.conf.tpl /root/default.conf.tpl
RUN npm i \
 && npm run dist

FROM nginx:1.15.0-alpine
LABEL MAINTAINER="Maksim Kostromin https://github.com/daggerok"
ARG PORT_ARG="80"
ARG HOST_ARG="localhost"
ENV PORT=${PORT_ARG}
ENV HOST=${HOST_ARG}
RUN apk add --update --no-cache libintl curl \
 && apk add --no-cache --virtual build_deps gettext \
 && cp /usr/bin/envsubst /usr/local/bin/envsubst \
 && apk del build_deps \
 && echo '#!/bin/sh' >> /usr/local/bin/entrypoint.sh \
 && echo 'PORT=${PORT:=80}' >> /usr/local/bin/entrypoint.sh \
 && echo 'HOST=${HOST:=localhost}' >> /usr/local/bin/entrypoint.sh \
 && echo '#/bin/sh -c "envsubst < /root/default.conf.tpl >
/etc/nginx/conf.d/default.conf && exec nginx -g \"daemon off;\""' >>
/usr/local/bin/entrypoint.sh \
 && echo '/bin/sh -c "envsubst < /root/default.conf.tpl >
/etc/nginx/conf.d/default.conf && (cat /etc/nginx/conf.d/default.conf | grep listen)
&& (cat /etc/nginx/conf.d/default.conf | grep server_name) && exec nginx -g \"daemon
off;\""' >> /usr/local/bin/entrypoint.sh \
 && chmod +x /usr/local/bin/entrypoint.sh
ENTRYPOINT entrypoint.sh
CMD /bin/bash
COPY --from=0 /root/default.conf.tpl /root/default.conf.tpl
COPY --from=0 /root/mean-stack/frontend/dist/frontend /usr/share/nginx/html
```

build everything run and test...

```
docker build --rm --no-cache -f Dockerfile.frontend-build -t daggerok/mean-stack-
frontend-build .
docker run -d --rm --name run-mean-stack-frontend-build -p 80:80 daggerok/mean-stack-
frontend-build
open http://0.0.0.0/
docker stop run-mean-stack-frontend-build
```

5.1. same, but build in heroku

Here, we wanna simply delegate everything to `git push heroku master` instruction

prepare `heroku.yml` *file:*

```
build:
  docker:
    web: Dockerfile.frontend-build
```

```
cp -Rf ./.gitignore.local ./.gitignore
cp -Rf ./heroku.yml.frontend-build ./heroku.yml
cp -Rf ./.dockerignore.frontend-build ./.dockerignore
git add .
git commit -am "Deploy heroku app."
git push heroku master
```

Chapter 6. Links

GitHub

asciidoctor reference...