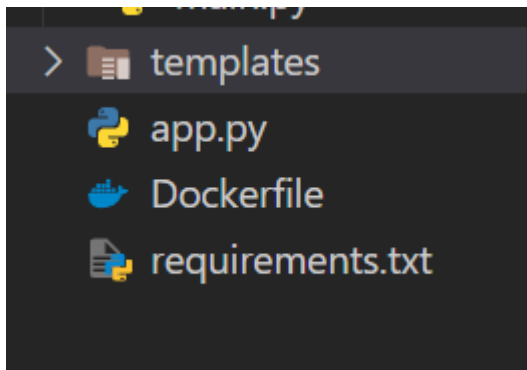


1.user registratin



```
app.py > ...
1  from flask import Flask,request,render_template
2  app=Flask(__name__)
3  @app.route("/register",methods=['GET','POST'])
4  def register():
5      if(request.method=='POST'):
6          name=request.form['name']
7          email=request.form['email']
8          password=request.form['password']
9          return render_template('success.html')
10     return render_template('register.html')
11 if __name__=='__main__':
12     app.run('0.0.0.0')
```

```
Dockerfile
1  FROM python:3.8
2  WORKDIR /app
3  COPY . .
4  RUN pip install --no-cache-dir -r requirements.txt
5  EXPOSE 5000
6  CMD ["python", "app.py"]
```

```

templates > register.html > html > body
 2  <html lang="en">
 3  <head>
 7  </head>
 8  <body>
 9      <form action="/register" method="post">
10          <input type="text" name='name' placeholder="name">
11          <input type="email" name="email" placeholder="email">
12          <input type="password" name="password" placeholder="password">
13          <button type="submit">Submit</button>
14      </form>
15  </body>
16  </html>

```

```

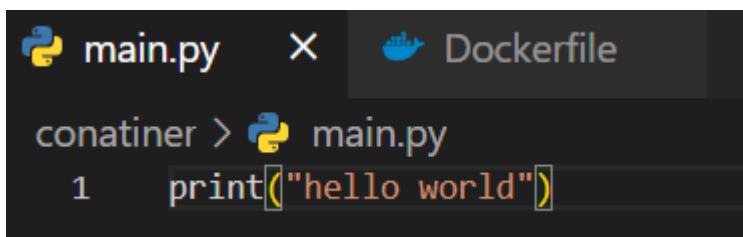
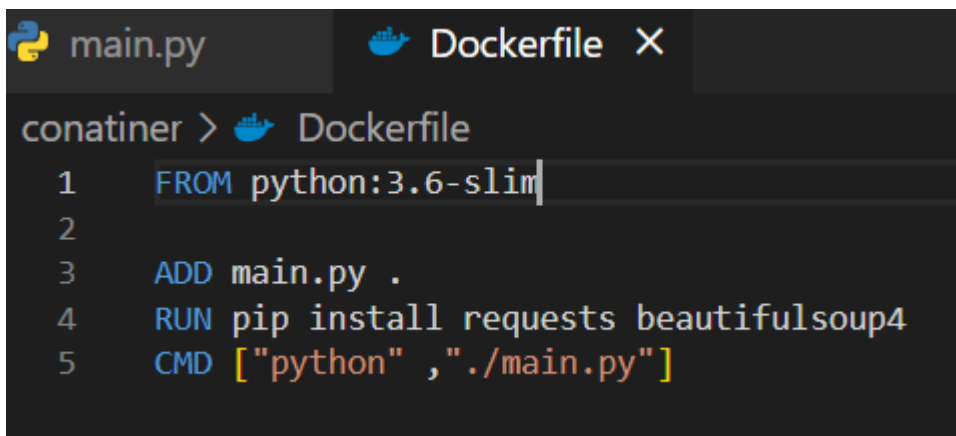
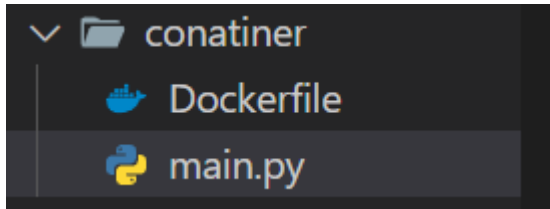
templates > success.html > html > body > h1
 1  <!DOCTYPE html>
 2  <html lang="en">
 3  <head>
 4      <meta charset="UTF-8">
 5      <meta name="viewport" content="width=device-width, initial-scale=1">
 6      <title>Document</title>
 7  </head>
 8  <body>
 9      <h1>Registration successful</h1>
10  </body>
11  </html>

```

docker build -t mainfolder-name

docker run -p 5000:5000 mainfolder-name

7.docker containerization



docker build -t folder-name .

docker run -t -i folder-name

6.docker commands

```
PS C:\Users\Admin> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" run --name mycontainer -it ubuntu:16.04 /bin/bash
Unable to find image 'ubuntu:16.04' locally
16.04: Pulling from library/ubuntu
58699f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:1f1a2d56de1d604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6
Status: Downloaded newer image for ubuntu:16.04
root@85703a9c67b9:/# docker start mycontainer
```

```
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" start mycontainer
>>
mycontainer
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" start mycontainer
>>
mycontainer
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" start mycontainer
>>
mycontainer
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" stop mycontainer
mycontainer
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" rm mycontainer
mycontainer
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
b6ccbedc98cf   repo1         "python main.py"        5 weeks ago   Exited (0) 5 weeks ago           quirky_wescoff
da9f165e849f   project1:latest "python main.py"        6 weeks ago   Exited (255) 5 weeks ago   8000/tcp   trusting_lehmann
f8707d0c09a6   project1:latest "python main.py"        6 weeks ago   Exited (255) 5 weeks ago   8000/tcp   confident_dirac
037e8d095b42   project1      "python main.py"        6 weeks ago   Exited (255) 5 weeks ago   0.0.0.0:8000->8000/tcp   nervous_cray
9ffe73401439   flask-docker-registration "python app.py"        7 weeks ago   Exited (0) 7 weeks ago           serene_bose
565b77b6a5ab   flask-docker-registration "python app.py"        7 weeks ago   Exited (0) 7 weeks ago           focused_mendel
119436788de8   flask-docker-registration "python app.py"        7 weeks ago   Exited (0) 7 weeks ago           elastic_euclid
e072d117354f   flask-docker-registration "python app.py"        7 weeks ago   Exited (0) 7 weeks ago           exciting_rubin
e4a4aefa5dd    flask-docker-registration "python app.py"        7 weeks ago   Exited (0) 7 weeks ago           xenodochial_jepse
n
```

```
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
repo1         latest    f143a8d45202   5 weeks ago    1.03GB
project1      latest    96fbc415afa2   6 weeks ago    1.03GB
flask-docker-registration latest    67ae4092955a   7 weeks ago    1.01GB
ubuntu        16.04     b6f507652425   2 years ago    135MB
PS D:\chs-java> & "C:\Program Files\Docker\Docker\resources\bin\docker.exe" pull ubuntu:16.04
16.04: Pulling from library/ubuntu
Digest: sha256:1f1a2d56de1d604801a9671f301190704c25d604a416f59e03c04f5c6ffee0d6
Status: Image is up to date for ubuntu:16.04
docker.io/library/ubuntu:16.04
```

: \$ docker run --name mycontainer -it ubuntu:16.04 /bin/bash

\$ docker start mycontainer

\$ docker stop mycontainer

: \$ docker rm mycontainer

\$ docker ps

\$ docker images

\$ docker pull ubuntu:16.04

\$ docker push myimage

```
3.github
git clone https://github.com/varsha-kadaru/Jenkins.git
git add .
git commit -m "mgs"
git push
```

SELENIUM

```
const {Key,until,Build,By}=require('selenium-webdriver');
(async function example(){

    let driver=await new Build().forBrowser('chrome').build();

    try{
        await driver.get('https://www.google.com');
        await driver.wait(until.titleContains("Google"),10000);
        console.log("testcase 1")
        await driver.findElement(By.name("q")).sendKeys("Selenium",Key.RETURN);
        await driver.wait(until.titleContains("Selenium"),1000);
        console.log("Testcase 2")
        const search=await driver.findElements(By.css('div.g'))
        console.log(`test case 3 found ${search.length} el`);
        const isSearch=await driver.findElement(By.name("q"));
        const isS=await isSearch.isDisplayed();
        console.log(`test case 4 passed ${isS}`)
    }catch(error){
        console.log("faile",error);
    }
    finally{
        console.log("Quit browse")
        await driver.quit();
    }

})();
```


1) User registration form.

create index.html, success.html } in templates folder

Dockerfile

main.py

requirements.txt

a single

folder

In terminal :-

python

main.py

2) Git & Github

→ Create a repo in github account

→ In VS Code create a folder with any file/files

→ From Github repo take the url.

In vs code :-

git clone -url-

then make connect path by
cd folder-name
then add repo name
cd repo-name

→ A folder with repo name will be created in explorer of vscode.

→ then Move ^{all} the file's into the repo folder.

→ ∴ the folder & repo named folder gets integrated.

→ then execute these commands in terminal.

git add .

git commit -m "msg"

git push

∴ All the files will get pushed to repo.

3) Source Code Management in GitHub

Reflection of changes in VSCode to repo :-

git add

git commit -m "msg"

git push

} in VSCode
terminal

Ref of changes in repo to VSCode

git pull in VS Code terminal

4) Jenkins Installation

→ Check for Java version. (only JDK-21 is valid)

→ if not there, install it

→ Go to website Jenkins (for download)
(download & deploy)

- Download Jenkins 2.461 for windows

After Installation, follow the instructions, (next - next →)

⊕ in b/w of that next-next

Select JDE-Q1.

→ finally click on finish.

⇒ local host 8080. (open in chrome)

It asks for password.

there'll be an url.

Open Command prompt then

cd .. (clear's)

Use cd & get the desired url.

(until secrets)

then, type

Code

→ vs code "opens"

then there'll be password.

→ paste that password in that website.

Asks for pluggin installat'ns ✓ (click)

→ Click on sign in as admin.

Dashboard Open ✓ (Installation Complete)

Create a folder Jenkin programme.
In which we have all Java pgm.

New item → Freestyle project → OK.

advanced → use custom workspace
(give path of directory)
(Jenkin programs)

In add
→ build step
(execute windows batch shell commands)

↓

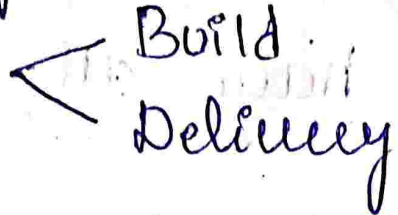
javae hi:java
java hi

↓
save.

click ← build now

↓
see the
console
output.

5) Continuous Integration & development using Jenkins.

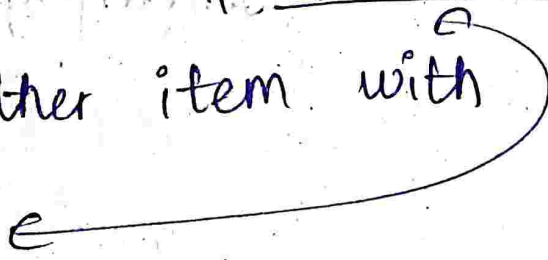
- i) executing 2 java files
- ii) Pipeline 
- iii) git integration

i) executing 2 java files.

Same steps as prev.

but add post build action.

& give a project name.

Now create another item with the project name 

Go to project 1 & build now:-

it gives o/p of project 1 &

gives link to project 2 output

ii) Pipeline

From Manage
Jenkin

→

Pluggins

↓
available

pluggins

↓
download

↓
build & delivery
pipeline.

→ Create a new view

→ Select build pipeline

→ Click on OK

Scroll down.

→ Select the initial project.

→ Select no. of builds (default 1)

→ Click on optional.

→ Click on OK leave

Pipeline is displayed ✓

For Delivery pipeline:

In add Components, use give initial
library

E. final to project's also
rest all process is same ✓.

iii) Crit Integration :-

→ Open github & go to a repository
which have a program file.

→ In that repository go to settings

→ Go to webhook.

↓
add new webhook

Go to other tab & search for "Know my IP".

Get that IP address &

⇒ In Payload URL enter -

http:// IP address /github-webhook

⇒ Change the Content type to
application/json

→ Under "which events would u like to trigger this webhook"

Click on "let me select individual events"

→ Click on add webhook ✓

Now go to jenkins:

New → freestyle
item project

Under source Code Management

Select "Git".

→ enter repository URL

& under credentials select anything other than none, if not present create one.

Under Branch specifier Change

* /master to * /main.

Under Build Triggers

Select "Github hook trigger for GITSCM polling"

then Under Build Steps:

type Commands for java file (like prev)

⇒ Click on save ✓

Click on build now → gives o/p.

6) Docker

Installation: Normally go to website & download.

i) for User registration

Dockerfile, main.py, requirements.txt & these files must be there.
templates → index.html
 → success.html

In terminal:-

Set the path using cd command.

then

Note: Docker must be running in background

docker build -t folder-name → compile

docker run -p 8080:8080 folder-name → run

✓ - 13

1785. J. W. W. W. W.

~~27-10-2024~~ ~~Angloport~~

2019

10/2/2017 5:20 PM

bin

62624

Coder
by
Varsha

George I. Smith

path) \rightarrow $\frac{1}{2} \log \frac{1}{2}$

DEC 30 1999

foldername

`-t img folder-name`

8, 9 → x

download

10) Install & explore Selenium

→ download Node.js

→ go to npm official website &

download Selenium web driver

(Choose firefox Component)

&

download

→ A zip file gets downloaded

geckodriver-win32.zip ✓

Go to C-drive & Create a folder

"Selenium webdriver"

& paste that applⁿ in zip folder

in this Selenium web driver

→ Open Command prompt from that folder

type: start geckodriver

other window opens.

if opens ✓ correctly installed
else wrong.

Now,

Create a folder "selen"

& open cmd from that folder &

enter `npm init -y`

next

`Code .`

vscode "ll" open.

In terminal,

`npm install selenium-webdriver`

→ In Selen, create index.js & enter
code.

In terminal :-

`node index.js`

will

→ You'll get output & a website[^] open
then selenium website opens & closes by
itself.