Week 7 - lab sheet - software setup, command line, and a template web application

In this lab sheet, we will:

Setup and install the IDE and command line software needed for the Web Programming course.

Become familiar with common command line tools to develop, understand and debug web applications.

Generate a template full-stack web application.

Setup and install the IDE and command line software

Please see the instructions here to install the needed software. There are instructions with scripts for Windows, Mac and Linux.

Ensure the version of jhipster is the 8.7.3 as in the given instructions.

Use the web command line tools

The command line is a powerful tool for any full stack developer. A full-stack developer is expected to be familar with some core command line tools and use these regualarly to develop, understand and debug web applications.

These tools will also help to understand how internet protocols work.

In a terminal or WSL/Ubuntu app if using Windows:

Exercise 1: Network Interface Configuration

Check your current network configuration

ifconfig

OR use the modern equivalent

ip addr show

Identify your:

- Local IP address (look for 192.168.x.x or 172.y.x.x (y=16 to 31) or 10.x.x.x)

- Public IP address (you may need to check your router or use curl ifconfig.me)

Questions:

What is your device's local IP address?

Can you identify which network interface is used for internet connectivity?

Is your IP address IPv4 or IPv6?

Exercise 2: DNS

Look up the IP address for google.com

nslookup google.com

Try a reverse lookup (IP to domain)

nslookup 8.8.8.8

Compare with dig command (if installed)

dig google.com

Check multiple record types

nslookup -type=MX gmail.com  # Mail exchange records

Questions:

What IP addresses does google.comLinks to an external site. resolve to?

What differences do you notice between nslookup and dig?

Why might a website have multiple IP addresses?

Exercise 3: Network Connectivity

Test basic connectivity to google.com

ping google.com

Limit to 5 packets

ping -c 5 google.com

Compare response times with different sites

ping -c 3 bham.ac.uk

ping -c 3 github.com

Questions:

What is the average response time to google.comLinks to an external site.?

Does packet loss indicate network problems?

Why might response times vary between different websites?

Exercise 4: Rest APIs

Test public REST APIs from the slides

curl https://pokeapi.co/api/v2/pokemon/ditto

curl "https://api.agify.io?name=madasar"

curl https://www.boredapi.com/api/activity

curl https://restcountries.com/v3.1/name/uk

Use jq to format JSON responses

curl https://pokeapi.co/api/v2/pokemon/ditto | jq

Questions:

What data format do these APIs return?

How can you modify the curl commands to get other data?

Find another useful RESTful API and use curl on it.

Exercise 5: Manual HTTP Requests with Telnet

Connect to a web server manually

telnet example.com 80

Then type these exact commands (type each line and press Enter):

GET / HTTP/1.1

Host: example.com

[Press Enter twice]

Exit with Ctrl+] then type 'quit'

Questions:

What response headers did you see?

How does this demonstrate that HTTP is a text-based protocol?

Why is understanding raw HTTP useful for debugging?

Exercise 6: More complex HTTP interactions with curl

Send a POST request with JSON data

```
curl -X POST https://httpbin.org/post \
-H "Content-Type: application/json" \
-d '{"name":"test","value":123}'
```

Send a request with custom headers

```
curl -H "User-Agent: MyCustomAgent/1.0" \
-H "Accept: application/json" \
https://httpbin.org/headers
```

Testing with Bearer token (simulated)

```
curl -H "Authorization: Bearer fake-token-123" \
https://httpbin.org/bearer
```

Testing with query parameters

```
curl "https://httpbin.org/anything?page=2&limit=10"
```

Upload a file with curl

```
echo '{"test": "data"}' > test.json
curl -X POST https://httpbin.org/post \
-F "file=@test.json" \
-F "name=testfile"
```

Questions:

How do headers affect server responses?

What is the purpose of the User-Agent header?

When would you need to send custom headers?

Exercise 7: SSH

Generate SSH key pair (if you don't have one)

ssh-keygen -t ed25519 -C "your_email@student.bham.ac.uk"

View your public key

cat ~/.ssh/id_ed25519.pub

Test SSH connection to a public server (like GitHub)

ssh -T git@github.com

Copy your public key to clipboard for server setup

cat ~/.ssh/id_ed25519.pub | clip.exe  # Windows WSL

OR

cat ~/.ssh/id_ed25519.pub | xclip -selection clipboard  # Linux

Questions:

Why use SSH keys instead of passwords?

What's the difference between public and private keys?

How do we get the command "ssh -T git@github.com" to show "Hi! You've successfully authenticated, but GitHub does not provide shell access."

Excercise 8: Docker web server

Run a simple web server in a container

docker run -d -p 8080:80 --name test-nginx nginx:alpine

Check running containers

docker ps

Inspect container network settings

docker inspect test-nginx | grep IPAddress

Test the container from host

curl http://localhost:8080

View container logs

docker logs test-nginx

Clean up

docker stop test-nginx && docker rm test-nginx

Questions:

How does port mapping work in Docker?

What is the difference between localhost and container IP?

Exercise N: additional advanced commands to research and try

Manually interact with HTTPS servers using openssl s_client like we did with HTTP via telnet

Check what process is using a specific port using lsof

Monitor network traffic in real-time using tcpdump

Generate a template full stack web application and run it

Using jhipster 8.7.3, generate a new full stack web application.

Using a terminal (WSL/Ubuntu app if using Windows) run the following commands to make a new direcory and change to that directory:

cd ~

mkdir -p git/dbwp

cd git/dbwp

jhipster

Then follow the prompts to create a new template application. Feel free to modify the themes and supported languages in the application.

shah@mars dbwp % jhipster



https://www.jhipster.tech

Welcome to JHipster v8.7.3

---

Documentation for creating an application is at https://www.jhipster.tech/creating-an-app/

Application files will be generated in folder: /Users/shah/git/dbwp

---

WARNING! _____

JHipster update available: 8.11.0 (current: 8.7.3)

Run npm install -g generator-jhipster to update.

---

✔️ What is the base name of your application? dbwp

✔️ Which type of application would you like to create? Monolithic application (recommended for simple projects)

✔️ What is your default Java package name? uk.ac.bham.cs

✔️ Would you like to use Maven or Gradle for building the backend? Maven

✔️ Do you want to make it reactive with Spring WebFlux? No

✔️ Which type of authentication would you like to use? JWT authentication (stateless, with a token)

✔️ Besides JUnit, which testing frameworks would you like to use?

✔️ Which type of database would you like to use? SQL (H2, PostgreSQL, MySQL, MariaDB, Oracle, MSSQL)

✔️ Which production database would you like to use? PostgreSQL

✔️ Which development database would you like to use? H2 with disk-based persistence

✔️ Which cache do you want to use? (Spring cache abstraction) Ehcache (local cache, for a single node)

✔️ Do you want to use Hibernate 2nd level cache? Yes

✔️ Which other technologies would you like to use?

✔️ Which framework would you like to use for the client? Angular

✔️ Besides Jest/Vitest, which testing frameworks would you like to use?

✔️ Do you want to generate the admin UI? Yes

✔️ Would you like to use a Bootswatch theme (https://bootswatch.com/)? Sketchy

✔️ Choose a Bootswatch variant navbar theme (https://bootswatch.com/)? Light

✔️ Would you like to enable internationalization support? Yes

✔️ Please choose the native language of the application English

✔️ Please choose additional languages to install Arabic (Libya), Chinese (Simplified), Hindi, Indonesian, Punjabi

Then try to run the application using:

./mvnw

This should download dependencies, build and run the web application.

Docker desktop may need to be running for ./mvnw to work.

Other things to try:

Use curl to access the application using http://localhost:8080/Links to an external site.

what do you notice about the HTML response to curl?

Open the application in you favourite web browser and login to the application

Explore the admin interface of the generated application - what are the main features provided in the template app?

Try to generate a skeleton application using a JDL file using the command jhipster jdl myapp.jdl

clone the repository from https://github.com/uob-team-project/team00Links to an external site. and run the given application

how is that application different from one you generated?