# BÀI THỰC HÀNH CHƯƠNG 1

**Họ và tên: Lê Thị Yên**

**MSSV: 20183861**

# 1. Web server apache2

## 1.1 Cài đặt web server apache2

Graphical user interface, application

Description automatically generated

**Câu 1:** Đường dẫn đến file html chứa nội dung mặc định của trang web Apache2 là:

/var/www/html/index.html

**Câu 2:** Cổng mặc định của dịch vụ www là 80

## 1.2 Cài đặt virtual hosts cho apache2

**Câu 3:** Với thự mục và file:

Chuỗi 755 cho biết thông tin phần quyền:

+ Owner: Đọc, Ghi, Thực thi ( 7 = 111(2) )

+ Group, Others: Đọc, Thực thi ( 5 = 101(2) )

**Câu 4:** Bạn quan sát thấy nội dung gì sau khi gõ 2 địa chỉ trên?

Graphical user interface, text, application

Description automatically generated

**Câu hỏi 5:** Thử truy cập từ các máy tính khác trong cùng mạng LAN vào 2 trang web đó.

Không truy cập được từ 1 máy tính trong mạng LAN

Graphical user interface, text, application, email

Description automatically generated

Sửa file hosts mới có thể truy cập vào được:

A screenshot of a computer

Description automatically generated with medium confidence

# 2. Interface trong Java

## 2.1 Xây dựng chương trình

**Câu 6:** Hãy tự viết một đoạn code để thực hiện 1 vòng lặp while sao cho nó sẽ nhận các số mà người dùng gõ và gửi về server, cho đến khi nào người dùng gõ ký tự rỗng rồi ấn enter.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Câu 7:** Vai trò của phương thức run là gì? Khi nào thì nó được gọi?

Vai trò của phương thức run là nhận dữ liệu từ clients, xử lý rồi gửi lại kết quả cho clients. Nó sẽ được gọi khi có 1 message từ client gửi đến server

# 3. Kiến trúc Microservices

**Câu 1:** Hãy thực hiện gõ những lệnh tương tự như trên với 3 dịch vụ còn lại.

**Câu 2:** Vào trang web DockerHub và đăng nhập vào tài khoản của bạn. Bạn thấy những gì mới xuất hiện trên docker hub repository của bạn?

Trên docker hub repository có 4 repository được push:

microservice-kubernetes-demo-customer

microservice-kubernetes-demo-order

microservice-kubernetes-demo-catalog

microservice-kubernetes-demo-apache

**Câu 3**: Trạng thái (status) của các pods vừa mới tạo được là gì? Bây giờ, hãy chờ vài phút và gõ lại lệnh đó, trạng thái mới của các pods giờ đã chuyển thành gì?

Graphical user interface

Description automatically generated

Trạng thái của các pods vừa mới được tạo là ContainerCreating

- Sau vài phút gõ lại lệnh, trạng thái các pods chuyển thành Running

Text

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

# 4. Kiến trúc JMS và DDS

**Câu 1:** Giải thích vai trò của application server glassfish.

Glassfish server là một application server mã nguồn mở được phát triền dựa trên nền JavaEE hỗ trợ đầy đủ các tính năng cần thiết của JavaEE như web ứng dụng, Enterprise JavaBeans, JPA, JavaServer Faces, JMS, RMI, JavaServer Pages, Servlets, etc. Nó cho phép những nhà phát triển ứng dễ quản lý, cấu hình nhanh và linh hoạt.

**Câu 3**: Sau khi chạy thử chương trình Sender và Receiver, vận dụng lý thuyết kiến trúc hướng sự kiện đã học trên lớp giải thích cơ chế chuyền và nhận thông điệp của Sender và Receiver.

Cơ chế chuyền và nhận thông điệpcủa Sender và Receiver:

* Sender:

1. Create and start connection
2. Create queue session
3. Get the Topic object
4. Create TopicPublisher object
5. Create TextMessage object
6. Write message
7. Send message
8. Connection close

* Receiver:

1. Create and start connection
2. Create topic session
3. Get the Topic object
4. Create TopicPublisher object
5. Create Listener object
6. Register the listener object with subscriber

**Câu 4:** So sánh JMS và DDS.

|  |  |
| --- | --- |
| JMS | DDS |
| * Java Messaging Service * Java Message Oriented Middleware (MOM) API * Used for sending messages between two or more clients * Centrailized * Allows the communication between different components of a distributed application * Most commonly used * Many implementations (ActiveMQ, GlassFish, RabbitMQ,....) | * Data distribution service * DDS is networking middleware that simplifies complex network programming * It implements a publish/subscribe model for sending and receiving data, events, and commands among the nodes * Decentralized * Automatically handles all aspects of message delivery * Automatically handles hot-swapping redundant publishers if the primary fails |

|  |
| --- |
|  |

* JMS:
  + Ưu điểm:

+ Độ tin cậy.

**+ Asynchronous** (Bất đồng bộ)

**+ Ease of Integration** (dễ tích hợp)

**+Scalability** (Khả năng mở rộng

**+ Flexibility** (Tính linh hoạt)

**+ Loosely coupled** (Kết nối lỏng lẻo)

+ **Interoperability** (Khả năng tương tác)

* + Nhược điểm: JMS sẽ không bao gồm các tính năng sau, bởi vì JMS chỉ là hệ thống gửi nhận tin:

+ Load Balancing/Fault Tolerance

+ Error/Advisory Notification

+ Administration

+ Security

+ Wire Protocol

+ Message Type Repository

* DDS:

Ưu điểm chính của DDS là nó kết hợp cả nhắn tin và bộ nhớ đêm dữ liệu để đáp ứng các yêu cầu khắt khe của các ứng dụng thời gian thực phân tán. Giao diện DDS được hỗ trợ trên nhiều nền tảng, bao gồm cả phần cứng nhúng chạy hệ điều hành thời gian thực

Tham khảo: <https://topdev.vn/blog/gioi-thieu-jms-java-message-services/>