

Báo cáo bài tập thực hành môn hệ thống phân tán và ứng dụng IT4611

Họ và tên: Lưu Tuấn Hùng

MSSV: 20225131

Nội dung báo cáo

Họ và tên sinh viên: Lưu Tuấn Hùng

MSSV: 20225131

Web server apache2

Câu hỏi 1: Đường dẫn đến file html chứa nội dung mặc định của trang web các bạn vừa xem là gì?

Đường dẫn: `/var/www/html/index.html`

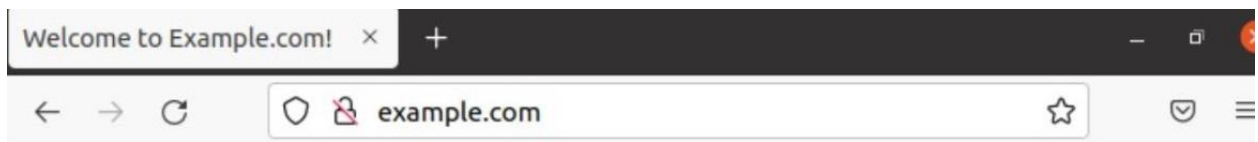
Câu hỏi 2: Cổng mặc định của dịch vụ www là gì?

Cổng mặc định của dịch vụ www là: 80

Câu hỏi 3: Hãy giải thích quyền mang số 755 là gì

Chủ sở hữu: $7 = 4 + 2 + 1 \Rightarrow$ Quyền ghi (4), quyền đọc(2), quyền thực thi hoặc truy cập(1)

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



Success! The example.com virtual host is working!



Success! The test.com virtual host is working!

Giải thích:

Cơ chế hoạt động:

- File /etc/hosts trỏ example.com và test.com về 127.0.0.1 (localhost).
- Apache2 nhận yêu cầu qua port 80, đọc file cấu hình (example.com.conf và test.com.conf) để xác định DocumentRoot.
- DocumentRoot trỏ đến /var/www/example.com/public_html (hoặc test.com/public_html), nơi chứa file index.html.
- Apache phục vụ file index.html, trình duyệt hiển thị nội dung HTML.

Nội dung hiển thị: File index.html trong /var/www/example.com/public_html

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

Nội dung không đổi so với câu trước

2. Interface trong Java

Câu hỏi 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. Gọi ý: hãy dùng lệnh sau để nhận xâu ký tự người dùng gõ vào: String message = scanner.nextLine();

```
StringBuilder stringBuilder = new StringBuilder();
```

```
while (true) {
```

```
    String message = scanner.nextLine();
```

```
    if (message.equals("")) {
```

```
        break;
```

```
}
```

```
    stringBuilder.append(message).append("\n");
```

```

}

out.println(stringBuilder.toString());

socket.close();

scanner.close();






















```

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






- Vai trò của phương thức run:
 - Tạo luồng vào ra cho server kết nối tới client
 - Gửi message Welcome tới client
 - Nhận dữ liệu từ client gửi tới
 - Xử lý việc sắp xếp các số mà client gửi tới
 - Gửi trả lại cho client mảng số sau khi đã sắp xếp
- Phương thức được gọi khi có một client kết nối tới server



3. Kiến trúc microservice

Câu 1:

<input type="checkbox"/>	Name	Tag	Image ID	Created	Size	Actions
<input type="checkbox"/>	 huhjhl75/microservice-k	latest	f7cd9e0f87cf	21 minutes ago	386.63 MB	  
<input type="checkbox"/>	 microservice-kubernetes	latest	f7cd9e0f87cf 	21 minutes ago	386.63 MB	  
<input type="checkbox"/>	 huhjhl75/microservice-k	latest	dcb0e1dc5bb6	3 minutes ago	492.59 MB	  
<input type="checkbox"/>	 microservice-kubernetes	latest	dcb0e1dc5bb6	3 minutes ago	492.59 MB	  
<input type="checkbox"/>	 huhjhl75/microservice-k	latest	2d0d0aed6afa	2 minutes ago	492.59 MB	  

Showing 5 items

Name	Last Pushed 	Contains	Visibility	Scout
huhjhl75/microservice-kubernetes-demo-order	2 minutes ago		Public	Inactive
huhjhl75/microservice-kubernetes-demo-customer	2 minutes ago		Public	Inactive
huhjhl75/microservice-kubernetes-demo-catalog	3 minutes ago		Public	Inactive
huhjhl75/microservice-kubernetes-demo-apache	6 minutes ago		Public	Inactive

1-4 of 4  

Câu 2: Tất cả đều ở trạng thái running

D:\2024.2\he_phan_tan\tuan_1\microservices-demo>kubectll get all

NAME	READY	STATUS	RESTARTS	AGE
pod/apache-5767bdbb87-k6nnj	1/1	Running	0	95m
pod/catalog-7556bb9bd8-rb9cs	1/1	Running	0	95m
pod/customer-79c567df85-jm22r	1/1	Running	0	95m
pod/order-7d8f986b8c-64qkw	1/1	Running	0	95m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/apache	LoadBalancer	10.96.141.92	<pending>	80:30569/TCP	95m
service/catalog	LoadBalancer	10.100.129.250	<pending>	8080:31450/TCP	95m
service/customer	LoadBalancer	10.100.225.233	<pending>	8080:31035/TCP	95m
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	98m
service/order	LoadBalancer	10.107.171.110	<pending>	8080:30353/TCP	95m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/apache	1/1	1	1	95m
deployment.apps/catalog	1/1	1	1	95m
deployment.apps/customer	1/1	1	1	95m
deployment.apps/order	1/1	1	1	95m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/apache-5767bdbb87	1	1	1	95m
replicaset.apps/catalog-7556bb9bd8	1	1	1	95m
replicaset.apps/customer-79c567df85	1	1	1	95m
replicaset.apps/order-7d8f986b8c	1	1	1	95m

Câu 3: Deploy

Order Processing

- Customer

Catalog

Catalog

Order
- List / add / remove customers

List / add / remove items

Search Items

Create an order

Customer : View all

id	Name	Firstname	
1	Tran	Hai Anh	delete
2	Nguyen	Thi Linh	delete
3	Dao	Van Tuan	delete
4	Luu Tuan Hung	Hung	delete

[Add Customer](#)

Item : View all

id	Name	Price	
1	iPod	42.0	delete
2	iPod touch	21.0	delete
3	iPod nano	1.0	delete
4	Apple TV	100.0	delete
5	samsung	10.7	delete

[Add Item](#)

Item : View all

id	Name	Price	
5	samsung	10.7	delete

[Add Item](#)

Order : View all

ID	Customer	Total Price	
1	Hung Luu Tuan Hung	21.4	delete

[Add Order](#)

1. Kiến trúc JMS và DDS

Câu hỏi 1:

GlassFish là một application server mã nguồn mở, hỗ trợ triển khai và quản lý các ứng dụng Java EE (Enterprise Edition). Trong bài thực hành này, GlassFish đóng vai trò quan trọng như sau:

- Message Broker trong JMS: GlassFish quản lý các kết nối và đích (topics/queues) trong mô hình Publish/Subscribe của JMS (Java Message Service). Nó đảm bảo thông điệp được gửi từ Sender đến Receiver một cách hiệu quả.
- Cung cấp giao diện quản lý: Thông qua giao diện web trên cổng 4848 (<http://localhost:4848>), GlassFish cho phép người dùng cấu hình và theo dõi các tài nguyên JMS như Connection Factories và Destinations.
Tóm lại, GlassFish là trung gian không thể thiếu để hỗ trợ giao tiếp bất đồng bộ trong hệ thống JMS.

Câu hỏi 2:

Trong bài thực hành, hai JNDI được tạo là myTopicConnectionFactory và myTopic. Lý do cần tạo hai JNDI này là:

- myTopicConnectionFactory: Đây là một Connection Factory, dùng để thiết lập kết nối từ client (Sender/Receiver) đến GlassFish. Nó cung cấp cơ chế quản lý các kết nối JMS, giúp các thành phần trong chương trình truy cập vào hệ thống nhắn tin.
- myTopic: Đây là một Destination (cụ thể là Topic trong mô hình Publish/Subscribe), đại diện cho điểm đến của các thông điệp. Sender gửi thông điệp đến Topic này, và Receiver nhận thông điệp từ đó.

Hai JNDI này cần thiết để các thành phần Sender và Receiver có thể tra cứu (lookup) thông qua JNDI và sử dụng chúng trong mã Java để gửi/nhận thông điệp một cách chuẩn hóa và tách biệt.

Câu hỏi 3:

Kiến trúc hướng sự kiện (Event-Driven Architecture - EDA) là một mô hình mà các thành phần giao tiếp thông qua việc phát và nhận sự kiện (thông điệp) thông qua một trung gian (broker), không cần biết chi tiết về nhau. Cơ chế hoạt động của Sender và Receiver trong chương trình được giải thích như sau:

- Sender (Publisher):

Gửi thông điệp đến Topic (myTopic) thông qua myTopicConnectionFactory. Sender không cần biết ai sẽ nhận thông điệp, thể hiện tính tách rời (decoupling) của EDA.

- Receiver (Subscriber):

Đăng ký nhận thông điệp từ Topic (myTopic). Khi có thông điệp mới, đối tượng MyListener sẽ được kích hoạt để xử lý thông điệp một cách bất đồng bộ.

- GlassFish (Broker):

Đóng vai trò trung gian, quản lý và phân phối thông điệp từ Sender đến tất cả Receiver đã đăng ký với Topic.

Giải thích theo EDA: Sự kiện (thông điệp) được gửi qua Topic mà không có giao tiếp trực tiếp giữa Sender và Receiver. Điều này giúp hệ thống linh hoạt, dễ mở rộng và giảm sự phụ thuộc giữa các thành phần.

Câu hỏi 4: So sánh JMS và DDS.

So sánh giữa JMS (Java Message Service) và DDS (Data Distribution Service):

JMS

- Định nghĩa: Là một API trong Java, hỗ trợ giao tiếp bất đồng bộ giữa các ứng dụng thông qua message broker.
- Mô hình: Hỗ trợ Point-to-Point (Queue) và Publish/Subscribe (Topic).
- Ưu điểm:

Dễ tích hợp trong các ứng dụng Java.

Hỗ trợ nhiều broker như GlassFish, ActiveMQ.

- Nhược điểm:

Phụ thuộc vào broker trung tâm, có thể gây độ trễ hoặc điểm lỗi duy nhất (single point of failure).

DDS

- Định nghĩa: Là tiêu chuẩn giao tiếp dữ liệu thời gian thực, hoạt động theo mô hình peer-to-peer (không cần broker).
- Mô hình: Publish/Subscribe với hỗ trợ QoS (Quality of Service) mạnh mẽ.
- Ưu điểm:

Phân tán hoàn toàn, không có điểm lỗi duy nhất.

Phù hợp với hệ thống thời gian thực như IoT, xe tự hành.

- Nhược điểm:

Phức tạp hơn, yêu cầu cấu hình QoS chi tiết.

So sánh chi tiết

- Kiến trúc: JMS: Dùng broker trung tâm (như GlassFish).

DDS: Peer-to-peer, không cần trung gian.

- Hiệu năng:

DDS nhanh hơn trong môi trường thời gian thực do loại bỏ broker.

JMS có thể chậm hơn do phụ thuộc vào broker.

- Dễ sử dụng:

JMS đơn giản hơn trong ứng dụng Java thông thường.

DDS phức tạp hơn nhưng mạnh mẽ và linh hoạt hơn cho hệ thống phân tán.