An- Najah National University

Faculty of Engineering

Department Name:
Computer Engineering

DosProject_Part2

Instructor Name: **Dr. Samer arande**
Student Name:**Jawna Amjad Atwe – Rahaf Ali Nawwas**
Academic Year:**2022/2023**

Replication and Caching

In this part, we will add replication and caching to improve request processing latency. While the front-end server in lab 1 was a very simple component, in this part, we will add two types of functionality to the front-end node. First, we will add an in-memory cache that caches the results of recent requests to the order and catalog servers.

Caching

I implement caching by adding hash table in front service which append any book into the table after info request, but if there is any updates on any book in the table catalog server will send a request to front service to delete it from hash table.

```
Hashtable<Integer,Book> hm=new Hashtable<Integer,Book>();
public Book info(int item) {
    chooseReplicaCatalog();
    if(hm.containsKey(item))
    {
        return hm.get(item);
    }
    try {
        Book book=restTemplate.getForObject("http://192.168.56.1:"+port+"/BookItem/"+item, Book.class);
        hm.put(book.getId(), book);
        return book;
    } catch (Exception e) {
        throw new ProductNotfoundException();
    }

    }
```

In this part of code front check if the book is in the cache it will take it from the hash table but if the hash table does not has the book it will send request to catalog then add the book to hash table.

```
public void updateStock(int id,String value, int calledFrom)
{
    try {
    if(value.equalsIgnoreCase("inc"))
    {

        book_Data.updateStockInc(id);
        restTemplate.getForObject("http://192.168.56.104:8084/BookIDinvalid/"+id, Book.class);
        if(calledFrom==0)
            restTemplate.put("http://192.168.56.1:"+8081+"/Book/"+id+"/stock/"+value+"/1",Book.class);

    }
    else if(value.equalsIgnoreCase("dec"))
    {

        int stock=book_Data.findByStock(id);
        if(stock!=0)
        {
            book_Data.updateStockDec(id);
            restTemplate.getForObject("http://192.168.56.104:8084/BookIDinvalid/"+id, Book.class);
            if(calledFrom==0)
                restTemplate.put("http://192.168.56.1:"+8081+"/Book/"+id+"/stock/"+value+"/1",Book.class);

        }
    }
```

This part of code from catalog when any updates done on ant book it will send a request to front server to delete this book from hash table.

Replication

both the order and catalog server are replicated - their code and their database files are replicated on multiple machines. To deal with this replication, the front end node needs to implement a load balancing algorithm that takes each incoming request and sends it to one of the replicas. So i use a flag each time it will choose one of the replica.

```
void chooseReplicaCatalog()
{
    if(flag==0)
    {
        port=8080;
        flag=1;
    }
    else {
        port=8081;
        flag=0;
    }
}
```