# 字节Data二面

## SpringMVC处理请求机制

> 当场死亡

## url从浏览器到服务端的一系列问题

### 域名解析迭代法的具体实现

### cookie、session、token

- token更安全吗？用什么加密算法？

  > 马马虎虎说了

### 常见请求

- PUT/GET/UPDATE/POST/PUT
- UPDATE是啥？
- GET一般用来做什么
- POST一般用来做什么

### 常见状态码

- 301 / 302 ？又说反了，还扯到转发和重定向....

## 排序

### 快排

- 算法复杂度
- 哪种情况下效率最低
- 如何提高效率

  > 答了可以通过随机选择基准，她问有没有其他的，不懂

### 希尔排序

- 算法复杂度
- 是不是稳定排序？

  > 又答反了...

## Git相关

- 常见指令
- 如何回滚一个版本

  > 不会

## MarkDown相关

- 常用语法

## 操作系统

### 进程与线程的概念和区别

### 堆栈在线程中是共享还是私有?

> 其实我说的是堆是共享的，栈是私有的，不知道对不对

### Linux用过吗?

> 用的少，会一些简单指令

## 算法题

> 1->2->3->4->5->6 单链表2个为一组翻转
>
> 5->6->3->4->1->2

```
import java.util.*;
public class Main {
    private static ListNode{
        int val;
        ListNode next;
    }
    public static ListNode solution(ListNode head){
        if(head == null) return head;
        int k = 2;
        ListNode ansTail = null;
        ListNode ansHead = null;
        ListNode cur = head;
        while(cur != null){
            ListNode end = cur;
            ListNode pre = cur;
            for(int i = 0; i < k && end != null; ++i){
                pre = end;
                end = end.next;
            }
            pre.next = ansTail;
            ansTail = pre;
            ansHead = cur;
            cur = end;
        }
        return ansHead;
    }
    public static void main(String[] args) {
        //Scanner in = new Scanner(System.in);
        //int a = in.nextInt();
        //System.out.println(a);
        System.out.println("Hello World!");
    }
}
```

# 生产者消费者模型

这是我当时的代码

```
#include <iostream>
using namespace std;
class BlockingQueue{
    private Queue<Integer> queue;
    public BlockingQueue(){
        queue = new LinkedList<Integer>();
    }
    public void generatorData(int val){
        synchronized(queue){
            queue.offer(val);
            try{
                queue.notifyAll();
            }catch(Exception e){
                //todo
            }
        }
    }
    public void buyData(){
        while(true){
            if(!queue.isEmpty()){
                synchronized(queue){
                    if(!queue.isEmpty()){
                        System.out.println(queue.poll());
                    }else{
                        queue.wait();
                    }
                }
            }
        }
    }
}
```