## 1.什么是分布式?

根据特定的需求,将大型项目进行拆分

## 2.项目拆分方式有哪些?

**垂直拆分:**将大型项目根据**功能模块(对外统一,对内独立)**进行拆分.

**水平拆分:**将大型项目按照**层级**进行拆分(Controller/Service/Mapper/Pojo/View)

## 3.什么是集群？

是抗击高并发最有效的一种手段

## 4.什么是高可用？

是分布式系统架构中必须要考虑的问题，通过设计来减少系统不能提供服务的时间

**一般用集群化来解决**

## 5.Maven如何保证数据的有效性？

① MD5 不可逆 加盐值

② SHA1 （安全哈希算法）

## 6.什么是SSM框架？

① Spring:用来整合其他的主流框架

1. IOC（控制反转）将对象创建交给第三方，需要使用时注入即可
2. AOP 代理（JDK/CGLIB）不修改源码的前提下，对方法进行扩展

② SpringMVC:就是用来接收用户请求，实现数据的响应

③ Mybatis:负责数据持久层与数据库进行交互（ORM）

## 7.什么是ORM？

对象关系映射

结果集 user\_id 去除\_后首字母大写 userId

## 8.SSM框架是如何搭建的？ WEB.XML

**首先配置Controller也就是SpringMVC:**

1. 配置前端控制器
2. 开启MVC注解
3. 视图解析器
4. 文件上传解析器

**Service 也就是 Spring**

1. 包扫描
2. Data source
3. 事物控制
4. 整合Mybatis

4-1） 管理SqlSessionFactory

4-2) 管理mapper接口

**Mapper 也就是 Mybatis**

1. 编辑Mybatis 配置
2. 编辑映射文件

## 9.什么是AOP？

面向切面编程，就是一种通过**预编译**和**运行期的动态代理**来实现程序统一维护的一种技术

AOP中默认的代理策略

如果目标对象有接口 默认采用JDK代理 **必须有接口**

如果目标对象没接口 采用CGLIB代理 创建出来的都是目标对象的子类

## 10.什么是JSON？

是一种轻量级的数据交互格式（字符串写法）

Object格式{key:value}

Array格式[key:value]

复杂格式 {key:value,key:[value,value]}

## 11.Mybatis中是如何多值传参的？

将多值封装为单值（对象/map集合/List）

## 12.什么是Nginx?

是一个高性能的HTTP和反向代理服务

特点：**反向代理、负载均衡**

策略：

1. 轮询策略
2. 权重策略：为性能高的处理器多分配
3. IP\_HASH策略
4. 服务优化策略

## 13.Linux的特点有什么？

1） 系统开源

2） 硬件要求低

3） 系统稳定性强

4） 安全性好

## 14.怎么实现数据库主从同步？

1） 主库有更新操作时，会被记录到主库的二进制文件中（**主库的二进制文件默认关闭**）

2） 从库通过IO线程读取二进制文件，并将信息存入中继文件中

3） 之后从库通过sql线程将中继文件中的数据取出最终同步到从库中

## 15.什么是双机热备？怎么实现？

为了实现多库数据的同步

可以使用Amoeba/Mycat等中间件实现

## 16.数据库的拆分方式有？

1） 垂直拆分:根据业务模块将DB拆分

2）水平拆分:将一个数据表的数据进行拆分

eg: A\_user ---> A\_user\_a A\_user\_b(水平)

## 17.数据库如何优化？

1） 优化SQL语句（**尽可能根据主键查询、单表查询**）

2） 创建索引、添加主键

3） 添加缓存（redis）在内存中

4） 提升硬件性能

5） 定期进行数据转储

6） 分库分表

## 18.什么是Redis?

内存中的数据结构存储系统，可以用作数据库缓存和消息中间件

## 19.什么是Redis分片？是如何存储的？

主要是为了内存的动态扩容

通过Hash一致性算法

根据节点IP算出地址 和 hash算法

如果出现负载不均，则引入虚拟节点

## 20.Redis如何实现持久化？

有两种方式RDB / AOF

区别：

AOF模式

实时备份

比RDB效率低

对原有文件追加操作

记录用户全部执行过程

内容明文不加密

RDB模式

定期备份

效率高

默认策略

内存的快照能够有效的节省空间

控制持久化文件大小

文件是加密的

## 21.lombok的用处。

可以自动生成POJO对象的全部方法

## 22.浏览器的跨域问题。

浏览器自带 **同源策略**

**访问协议://域名:端口**（三个都相同表示同源）

我们可以用JSONP和HttpClient两种方式解决跨域问题

## 23.简单介绍HttpClient和 JSONP以及区别。

JSONP是用来解决主流浏览器跨域问题的（域名）

HttpClient可以通过java代码访问任意的地址的数据

区别：

1. 请求发起的位置不同

JSONP是浏览器解析Ajax发起的请求

HttpClient是JAVA代码中模拟发起的请求

1. 安全性不同

JSONP浏览器可以记录JSONP的全部请求过程

HttpClient 因为请求是经过2次处理的所以相对安全

1. 请求方式不同

JSONP发起get请求，一般以查询为主

HttpClient 可以发起get/post请求

1. 代码调用层级不同

JSONP 3级调用

HttpClient 5级调用

## 24.SSO单点登录的流程。

1） 当用户点击登录按钮时，携带账户密码的请求发送给前台

2） 前台接收请求后，将请求封装，通过HttpClient中的POST请求将数据发送给SSO单点登录系统

3） 单点登陆系统接收到前台的数据后，通过数据库查询检验用户数据是否正确

不正确直接报错201

如果正确 则通过加密算法生成秘钥TOKEN

将用户数据转化为JSON

1. 将TOKEN和JSON保存到Redis中
2. 将后台生成的TOKEN返回给前台
3. 前台接收数据后通过Request对象操作Cookie将用户TOKEN保存到Cookie中
4. 登陆成功

## 25.用户注册流程。

1） 当用户点击注册按钮时，进行form表单的post提交

2） 前台Controller接收页面提交的全部参数

3） 接收数据后，进行业务处理，通过HttpClient、post提交实现数据远程传输

4） 后台通过Controller接收数据

5） 之后进行业务处理，最终入库保存，并且返回数据回执，告知前台入库情况

6） 前台接收后台的回执，进行判断如果是200则正确跳转如果是201 则友好提示报错返回

## 26.Session的共享问题。

**一定不共享**

通过拦截器 用户每次操作，都能获取用户信息，而用户信息通过request请求携带所以后续操作都能获取用户信息

## 27.什么是ThreadLocal？

可以在一个线程的任意位置，动态的获取数据（线程安全的）

## 28.什么是SOA思想？

微服务程序编程公式

面向服务的架构是一个组件模型，就是讲应用的不同功能单元（这里成为服务）进行拆分，并通过这些服务定义接口和契约联系起来，接口采用中立的方式进行定义的，它独立于硬件平台、操作系统和编程语言。使得构建在各种各样系统中的服务可以以一种统一和通用的方式进行交互。

## 29.微服务的特点是什么?

1. 分布式
2. 能够独立运行的
3. 能够自动化故障迁移

## 30.什么是OSI模型？

文件传输、电子邮件、文件服务、虚拟终端 HTTP/FTP/DNS/TELENT

数据格式化、代码转化、数据解密

解除或建立与其他接点的联系 RPC

提供端对端的接口 TCP/UDP

为数据包选择路由 IP/IMCP

传输有地址的帧，错误检测功能 ARP

以二进制数据形式在屋里没接上传输数据

应用层

表示层

会话层

传输层

网络层

数据链路层

物理层

## 31.http和RPC的区别？

1） 安全性

http更加安全，经过表示层加密

三次握手:目的保证数据传输的有效性

**http必须经过三次握手**

RPC直接发送数据，数据正确性不能保证

RPC内部，对数据进行加密

2）传输速度

RPC更快，是第5层协议

RPC采用长链接的形式

## 32.什么是SpringCloud?

为微服务带来**弹性、可靠性、和协调**简化了分布式微服务架构

## 33.什么是负载均衡？

说白了就是将一个任务分摊到多个操作单元上进行执行