**1.什么是java序列化？如何实现java序列化？**

Java 串行化技术可以使你将一个对象的状态写入一个Byte 流里，

并且可以从其它地方把该Byte 流里的数据读出来，重新构造一个相同的对象。

**2.jsp的内置对象？**

request response out session application pageContext page config exception

**3.oracle回滚机制**

回滚段用于存放数据修改之前的值（包括修改之前的位置和值）。一个事物只能使用一个回滚段来存放他的回滚信息，

一个回滚段可以存放多个事物的回滚信息

作用：当事物正在处理的时候，例程序失败，回滚段的信息保存在重做日志文件中，oracle将在下次打开数据库时利用回滚

来恢复未提交的数据

读一致性：当一个回话正在修改数据是，其他的回话将看不到该回话未提交的修改。而且，当一个语句正在执行时，该语句将

看不到从该语句开始执行后的未提交的修改（语句级读一致性）。

**4.** **Get和Post的区别**

1.get是从服务器上获取数据，post是向服务器传送数据，

2.get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。

3.get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

4.在进行文件上传时只能使用post而不能是get。

**5.重载，重写是什么**

**重载**：重载发生在同一个类中，在该类中如果存在多个同名方法，但是方法的参数类型和个数不一样，那么说明该方法被重载了。

**重写**：重写发生在子类继承父类的关系中，父类中的方法被子类继承，方法名，返回值类型，参数完全一样，但是方法体不一样，那么说明父类中的该方法被子类重写了。

**6.Hashtable与HashMap的区别**

HashMap不是线程安全的，HashTable是线程安全。

HashMap允许空（null）的键和值（key），HashTable则不允许。

HashMap性能优于Hashtable。

Map

1.Map是一个以键值对存储的接口。Map下有两个具体的实现，分别是HashMap和HashTable.

2.HashMap是线程非安全的，HashTable是线程安全的，所以HashMap的效率高于HashTable.

3.HashMap允许键或值为空，而HashTable不允许键或值为空.

**7.AJAX理解**

AJAX 全称： 异步JavaScript及 XML（Asynchronous JavaScript And XML）

Ajax的核心是JavaScript对象XmlHttpRequest(XHR)。

Ajax的优点:

　　　　提高用户体验度(UE)

　　　　提高应用程序的性能

　　　　进行局部刷新

AJAX不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术。

通过 AJAX，我们的 JavaScript 可使用JavaScript的XMLHttpRequest对象来直接与服务器进行通信。通过这个对象，我们的 JavaScript 可在不重载页面的情况与Web服务器交换数据，即可局部刷新。

AJAX 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面，减轻服务器的负担，提升站点的性能。

AJAX 可使因特网应用程序更小、更快，更友好，用户体验（UE）好。

Ajax是基于标准化并被广泛支持的技术，并且不需要插件和下载小程序

**8.冒泡排序**

public class Sort {

public static void sort() {

Scanner input = new Scanner(System.in);

int sort[] = new int[10];

int temp;

System.out.println("请输入10个排序的数据：");

for (int i = 0; i < sort.length; i++) {

sort[i] = input.nextInt();

}

for (int i = 0; i < sort.length - 1; i++) {

for (int j = 0; j < sort.length - i - 1; j++) {

if (sort[j] < sort[j + 1]) {

temp = sort[j];

sort[j] = sort[j + 1];

sort[j + 1] = temp;

}

}

}

System.out.println("排列后的顺序为：");

for(int i=0;i<sort.length;i++){

System.out.print(sort[i]+" ");

}

}

public static void main(String[] args) {

sort();

}

}

**9.阶乘**

public class Multiply {

public static int multiply(int num) {

if (num < 0) {

System.out.println("请输入大于0的数！");

return -1;

} else if (num == 0 || num == 1) {

return 1;

} else {

return multiply(num - 1) \* num;

}

}

public static void main(String[] args) {

System.out.println(multiply(10));

}

}

**10.jvm的内存结构**

java虚拟机的内存结构分为堆(heap)和栈(stack),堆里面存放是对象实例也就是new出来的对象。栈里面存放的是基本数据类型以及引用数据类型的地址。

对于所谓的常量是存储在方法区的常量池里面。

**11.单例**

单例就是该类只能返回一个实例。

单例所具备的特点：

1.私有化的构造函数

2.私有的静态的全局变量

3.公有的静态的方法

单例分为懒汉式、饿汉式和双层锁式

饿汉式:

public class Singleton1 {

private Singleton1() {};

private static Singleton1 single = new Singleton1();

public static Singleton1 getInstance() {

return single;

}

}

懒汉式：

public class Singleton2 {

private Singleton2() {}

private static Singleton2 single=null;

public tatic Singleton2 getInstance() {

if (single == null) {

single = new Singleton2();

}

return single;

}

}

线程安全：

public class Singleton3 {

private Singleton3() {}

private static Singleton3 single ;

public static Singleton3 getInstance() {

if(null == single){

synchronized(Singleton3.class){

if(null == single){

single = new Singleton3();

}

}

}

return single;

}

}

**12.解析xml文件的几种技术**

解析xml的几种技术

1.dom4j 2.sax 3.jaxb 4.jdom 5.dom

1.dom4j

　　 dom4j是一个Java的XML API，类似于jdom，用来读写XML文件的。dom4j是一个非常优秀的Java XML API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件。

2.sax

SAX（simple API for XML）是一种XML解析的替代方法。相比于DOM，SAX是一种速度更快，更有效的方法。它逐行扫描文档，一边扫描一边解析。而且相比于DOM，SAX可以在解析文档的任意时刻停止解析，但任何事物都有其相反的一面，对于SAX来说就是操作复杂。

3.jaxb

JAXB（Java Architecture for XML Binding) 是一个业界的标准，是一项可以根据XML Schema产生Java类的技术。该过程中，JAXB也提供了将XML实例文档反向生成Java对象树的方法，并能将Java对象树的内容重新写到XML实例文档。从另一方面来讲，JAXB提供了快速而简便的方法将XML模式绑定到Java表示，从而使得Java开发者在Java应用程序中能方便地结合XML数据和处理函数。

2、dom4j 与 sax 之间的对比：【注：必须掌握！】

dom4j不适合大文件的解析，因为它是一下子将文件加载到内存中，所以有可能出现内存溢出，

sax是基于事件来对xml进行解析的，所以他可以解析大文件的xml

也正是因为如此，所以dom4j可以对xml进行灵活的增删改查和导航，而sax没有这么强的灵活性

所以sax经常是用来解析大型xml文件，而要对xml文件进行一些灵活（crud）操作就用dom4j

**13.项目的生命周期**

1.需求分析

2.概要设计

3.详细设计(用例图，流程图，类图)

4.数据库设计(powerdesigner)

5.代码开发（编写）

6.单元测试（junit 白盒测试）(开发人员)

svn版本管理工具(提交，更新代码，文档)

7.集成测试 （黑盒测试，loadrunner（编写测试脚本）(高级测试)）

8.上线试运行 （用户自己体验）

9.压力测试（loadrunner）

10.正式上线

11.维护

**14.session和cookie的区别**

session是存储在服务器端，cookie是存储在客户端的，所以安全来讲session的安全性要比cookie高，然后我们获取session里的信息是通过存放在会话cookie里的sessionid获取的。又由于session是存放在服务器的内存中，所以session里的东西不断增加会造成服务器的负担，所以会把很重要的信息存储在session中，而把一些次要东西存储在客户端的cookie里，然后cookie确切的说分为两大类分为会话cookie和持久化cookie，会话cookie确切的说是存放在客户端浏览器的内存中，所以说他的生命周期和浏览器是一致的，浏览器关了会话cookie也就消失了，然而持久化cookie是存放在客户端硬盘中，而持久化cookie的生命周期就是我们在设置cookie时候设置的那个保存时间，然后我们考虑一问题当浏览器关闭时session会不会丢失，从上面叙述分析session的信息是通过sessionid获取的，而sessionid是存放在会话cookie当中的，当浏览器关闭的时候会话cookie消失所以我们的sessionid也就消失了，但是session的信息还存在服务器端，这时我们只是查不到所谓的session但它并不是不存在。那么，session在什么情况下丢失，就是在服务器关闭的时候，或者是sessio过期，再或者调用了invalidate()的或者是我们想要session中的某一条数据消失调用session.removeAttribute()方法，然后session在什么时候被创建呢，确切的说是通过调用session.getsession来创建，这就是session与cookie的区别

**15．final,finally,finalize 三者区别**

Final是一个修饰符：

当final修饰一个变量的时候，变量变成一个常量，它不能被二次赋值

当final修饰的变量为静态变量（即由static修饰）时，必须在声明这个变 量的时候给它赋值

当final修饰方法时，该方法不能被重写

当final修饰类时，该类不能被继承

Final不能修饰抽象类，因为抽象类中会有需要子类实现的抽 象方法，（抽 象类中可以有抽象方法，也可以有普通方法，当一个抽象类中没有抽象方 法时，这个抽象类也就没有了它存在的必要）

Final不能修饰接口，因为接口中有需要其实现类来实现的方法

Finally：

Finally只能与try/catch语句结合使用，finally语句块中的语句一定会执行， 并且会在return，continue，break关键字之前执行

finalize：

Finalize是一个方法，属于java.lang.Object类，finalize()方法是GC （garbage collector垃圾回收）运行机制的一部分，finalize()方法是在 GC清理它所从 属的对象时被调用的

**框架**

## Struts2的运行原理

1. 1、tomcat 启动的时候会加载 web.xml 、核心控制器 FilterDispatcher 会加载并解析 struts.xml
2. 2、客户端会发送一个请求到 action 、FilterDispatcher 会根据后缀名进行拦截
3. 3、FilterDispatcher根据 struts.xml 的配置文件信息 找到 某个action 对应的某个类里的指定方法
4. 4、执行相关的业务逻辑最后返回 一个String
5. 5、<action/> 里配置 <result/> name的属性值与返回的String 进行匹配,跳转到指定的jsp 页面

## 二、Spring MVC运行原理

整个处理过程从一个HTTP请求开始：

1.Tomcat在启动时加载解析web.xml,找到spring mvc的前端总控制器DispatcherServlet,并且通过DispatcherServlet来加载相关的配置文件信息。

2.DispatcherServlet接收到客户端请求，找到对应HandlerMapping，根据映射规则，找到对应的处理器（Handler）。

3.调用相应处理器中的处理方法，处理该请求后，会返回一个ModelAndView。

4.DispatcherServlet根据得到的ModelAndView中的视图对象，找到一个合适的ViewResolver（视图解析器），根据视图解析器的配置，DispatcherServlet将要显示的数据传给对应的视图，最后显示给用户。

## Spring MVC、struts1和struts2区别

1.spring mvc 单例 非线程安全

struts1单例 非线程安全

struts2线程安全对每个请求都产生一个实例

2.spring mvc的入口是servlet，而struts2是filter

spring 的前端总控制器为 DispatcherServlet

struts2 的前端总控制器为 FilterDispatcher

struts1 的前端总控制器为 actionServlet

3. 参数传递：struts是在接受参数的时候，

可以用属性来接受参数，这就说明参数是让多个方法共享的。

springmvc 用方法来接受参数

4.spring mvc是基于方法的设计,而sturts是基于类

**四、SSI整合**

1、Action继承于Actionsupport

2、引入struts-spring-plugin.jar包，从而完成struts和spring的整合

3、在struts2的action中注入service，保证service的名字和配置文件中的一致，并生成get,set方法

1. Dao层继承于SqlMapClientDaoSupport

5、在dao层的配置文件中注入sqlMapClient

**五、SSH整合**

1、Action继承于ActionSupport

2、引入struts-spring-plugin.jar包，从而完成struts和spring的整合

3、在struts2的action中注入service，保证service的名字和配置文件中的一致， 并生成get,set方法

4、Dao层继承于hibernateDaoSupport

5、在dao层的配置文件中注入sessionFactory

Hibernate中的两大配置文件：

　　　　\*.hbm.xml：主键生成策略，映射关系，一对多，一对一的关系

　　　　Hibernate.cfg.xml：方言(用哪个数据库)，数据库连接信息，包含\*.hbm.xml 内容，映射文件，也可以配事务

**六、Spring MVC整合**

1.首先,要在web.xml里面配置SpringMVC的核心控制器,DispatcherServlet,对指定的后缀请求进行拦截。

2.Controller层要加 @Controller注解,表明该类是MVC的控制层。

3.创建Service接口,给接口加上注解 @Component或者 @Service 表明这是Service业务处理层

4.在Controller层声明Service变量(属性),给变量（属性） 加上 @Autowired注解,通过自动绑定机制将Service注入到Controller。 (**注**：@Autowired默认是ByType,如果想根据属性名注入,那么就再加上注解 @Resource(name="属性名"))

5.在Controller层的方法上加上注解 @RequestMapping("requestAddress") 表明该方法的请求地址

6.Dao层要加上注解 @Repository 表明这是数据库持久层

7.同样将dao实例注入到service层中。

8.配置视图解析器 "InternalResourceViewResolver",对处理后的跳转进行统一配置。

**七、Hibernate 中get 和 load的区别**

加载方式：

　　　　load为延迟加载(返回的是一个只有id属性的代理,只有使用该对象属性时,才 发出sql语句)；

　　　　get为立即加载(执行时,会立即向数据库发出sql语句)

返回结果：

　　　　load检索不到记录时,会抛ObjectNotFoundException异常

　　　　get检索不到记录时,会返回null

**八、Hibernate、Ibatis、Jdbc三者的区别**

Hibernate属于全自动， Ibatis属于半自动，Jdbc属于手动，从开发效率上讲hibernate较高，ibatis居中，jdbc较低，从执行效率上讲hibernate较低，ibatis居中，jdbc较高，因为jdbc是手工写sql语句，程序员对sql的控制能力更大，可以根据业务需要进行优化，而ibatis虽然也可以对sql进行优化，但是他里面将resultset封装为实体的过程中采用了反射机制所以一定程度上影响了性能，而hibernate因为高度封装所以开发效率相对较高，但正因为这个原因，所以程序员在对sql语句的控制和优化方面相对比较弱，而且在将resultset封装成实体的过程中也采用了反射机制，所以在性能方面较低

1. **Hibernate的运行原理**

在初始化Hibernate时，会首先会根据configuration接口去加载并读取Hibernate的配置信息(hibernate.cfg.xml)，(一个Configeration实例代表Hibernate所有Java类到Sql数据库映射的集合。)，然后创建SessionFactory实例，把Configeration对象中的所有配置信息拷贝到SessionFactory的缓存中，最后通过调用SessionFactory创建Session对象，通过Session接口提供的各种方法来操纵数据库的访问(Query和Criteria接口)。

1. **Hibernate五大核心（类/接口）简述**

1 .Configuration接口的作用是对Hibernate进行配置，以及对它进行启动。（加载 hibernate.cfg.xml）并创建一个SessionFactory对象。

2 .SessionFactory接口

SessionFactory接口负责初始化Hibernate。它充当数据存储源的代理，并负责创建 Session对象。SessionFactory是线程安全的。

3 .Session接口

Session（会话）接口是Hibernate应用使用的主要接口。Session接口负责执行被持久化对象的CRUD操作(增删改查)。Session对象是非线程安全的。Session 相当于jdbc的connection

4 .Query与Criteria接口

总之Query和Criteria接口负责执行各种数据库查询。它可以使用HQL语句或SQL 语句两种表达方式。

1. .Transaction接口

Transaction（事务）接口是一个可选的API。负责操作相关的事务

**十一、Hibernate与JDBC的区别**

1. hibernate和jdbc主要区别就是，hibernate先检索缓存中的映射对象( 即hibernate操作的是对象)，而jdbc则是直接操作数据库.
2. Hibernate是JDBC的轻量级的对象封装，它是一个独立的对象持久层框架。Hibernate可以用在任何JDBC可以使用的场合
3. Hibernate是一个和JDBC密切关联的框架，所以Hibernate的兼容性和JDBC驱动，和数据库都有一定的关系，但是和使用它的Java程序，和App Server没有任何关系，也不存在兼容性问题。

4、如果正确的使用JDBC技术,它的执行效率一定比hibernate要好,因为hibernate是基于jdbc的技术.

5、JDBC使用的是SQL语句，Hibernate使用的是HQL语句，但是HQL语句最终还会隐式转换成SQL语句执行。

**十二、hibernate缓存概述**

hibernate分为：

**一级缓存**（session级别缓存）也叫事务级别的缓存

**二级缓存**（sessionFactory缓存）,也叫应用级缓存

**三级缓存**（查询缓存）

一级缓存的生命周期和session的生命周期保持一致，hibernate默认就启用了一级缓存，不能将其关闭，可以通过session.clear()和session.evict(object)来管理一级缓存。

其中get,load,iterate都会使用一级缓存，一级缓存缓存的是对象。

二级缓存的生命周期和sessionFactory的生命周期保持一致，可以跨session,被多个session共享，hibernate3默认开启二级缓存,也可以手动开启并指定缓存插件如ehcache,oscache等。二级缓存也只能缓存对象。

三级缓存也叫查询缓存，查询缓存是针对普通属性结果集的缓存,

对实体对象的结果集只缓存id。对query.list()起作用，query.iterate不起作用，也就是query.iterate不使用查询缓存

# 数据库篇

## JDBC连接数据库步骤(以MYSQL为例)

1、加载JDBC驱动程序：

通过Class类的forName方法实现，并将驱动地址放进去

成功加载后，会将Driver类的实例注册到DriverManager类中。

2、提供JDBC连接的URL 、创建数据库的连接

•要连接数据库，需要向java.sql.DriverManager请求并获得Connection对象，

该对象就代表一个数据库的连接。

•使用DriverManager的getConnectin()方法传入指定的欲连接的数据库的路径、数 据库的用户名和密码。

Connection con=DriverManager.getConnection(url , username , password);

&&&:"jdbc:mysql://localhost/test?user=root&password=123&useUnicode=true&characterEncoding=utf-8”;

3、创建一个Statement

•要执行SQL语句，必须获得java.sql.Statement实例

•执行静态SQL语句。通常通过Statement实例实现。

•执行动态SQL语句。通常通过PreparedStatement实例实现。

String sql = “”;

Statement st = con.createStatement() ;

PreparedStatement pst = con.prepareStatement(sql) ;

4、执行SQL语句

Statement接口提供了executeQuery、executeUpdate、execute三种方法

executeQuery：执行select语句，返回ResultSet结果集

ResultSet rst = pst.executeQuery();

• executeUpdate：执行insert、update、delete语句

pst.executeUpdate();

5、关闭JDBC对象

操作完成以后要把所有使用的JDBC对象全都关闭，以释放JDBC资源。

## 数据库连接池

数据库连接池简介：

数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接，当需要建立数据库连接时，只需从“缓冲池”中取出一个，使用完毕之后再放回去。

什么是数据库连接池：

Connection pooling，它是程序启动时建立足够的数据库连接，并将这些连接组成一个连接池，由程序动态地对池中的连接进行申请，使用，释放。

数据库连接池的运行机制：

1.从连接池获取或创建可用的连接；

2.使用完毕之后，把连接返还给连接池；

3.在系统关闭前，断开所有连接并释放连接占用的系统资源；

4.还能够处理无效连接（原来登记为可用的连接，由于某种原因不再可用，如超时、通讯问题），并能限制连接池中的连接总数不低于某个预定值和不超过某个预定值；

## mysql的数据库导入导出

配置：

首先找到mysql的安装目录，进入bin目录下复制路径

将mysql的bin目录粘贴在计算机环境变量的path中

授权：

登录mysql

将某张表的某个权限赋给某个用户

grant [select,insert,update,delete,create,drop] on [databaseName].[tableName] to [userName]@[userIP] identified by [‘连接口令’]

grant select,insert,update,delete,create,drop on oa\_ssh.user to root@[IP] identified by 'root';

将所有库的所有权限赋给某个用户

grant all privileges on \*.\* to [userName]@[userIp] identified by [‘连接口令’]

grant all privileges on \*.\* to root@[IP] identified by ‘root';

将所有库的所有权限赋给所有用户

　　　　grant all privileges on \*.\* to root@'%' identified by ‘root’;

　　　　导出本地数据库：

　　　　mysqldump -u 用户名 -p 数据库名 > 磁盘：导出的文件名(加后缀)

　　　　远程导出数据库：

　　　　mysqldump -h IP -u 用户名 -p 数据库名称 >导出的文件名(加后缀)

　　　　远程导出数据表：

　　　　mysqldump -u root -p -d --add-drop-table 数据库名称 > 导出文件

名(加后缀)

　　　　导入数据：

　　　　mysql -u root -p登录成功后 ==》 source 磁盘：导入的文件名(加后缀)

## jdbc分段批量提交的时候出现异常怎么处理?

通过Map来解决性能问题。首先在分段批量提交的时候，我们不采用事务，这样就保证了合法的数据就自动提交，不合法的数据就自己自动进行回滚，为了避免不合法数据影响后续合法数据的提交，采用定义业务规则字典表，实现对数据的验证，将不合法的数据记录下来，供用户进行后续处理，而合法的数据就全部提交。

## jdbc批量处理数据

dbc批量处理数据是通过PreparedStatement对象的 addbatch(), executebatch（） clearbatch()进行和数据库的交互。通常我们使用分段批量处理的方式 这样可以提高程序的性能 ，防止内存溢出。

　1.每个sql语句都和数据库交互一次(非批量操作)

　2.只和数据库交互一次(批量操作)(内存溢出)

　当数据达到一定额度的时候就和数据库进行交互，分多次进行(分段批量操作)

　 (500或者1000)

pst.addBatch();

if (i > 0 && i%1000 == 0) {

pst.executeBatch();

pst.clearBatch();

}

## Oracle分页

select \* from (select \* from (select s.\*,rownum rn from student s ) where rn<=5) where rn>0

## Oracle的基本数据类型

Oracle的基本数据类型（常用）：

1、字符型

Char 固定长度字符串 占2000个字节

Varchar2 可变长度字符串 占4000个字节

Nvarchar2 占2000个字符（最多能存2000个字母/中文）

2、大对象型（lob）

Blob ：二进制数据 最大长度4G

Blob 用于存一些图片，视频，文件。

比如：当我们在进行文件上传时，我们一般把上传的文件存在硬盘上，可以不占用 数据库，下载时，如果项目迁移时，文件也要跟着迁移。因此我们可以把用blob把它存在数据库中。但这样也增加了数据库的负担。

Clob ：字符数据 最大长度4G，可以存大字符串 varchar2和nvarchar2都具有一定的局限性，它们长度有限，但数据库中无论用varchar2或nvarchar2类型，还是用clob，在java端都使用String接收。

3、数值型

Integer 整数类型，小的整数。

Float 浮点数类型。

Real 实数类型。

Number（p,s）包含小数位的数值类型。P表示精度，s表示小数后的位数。

Eg: number(10,2) 表示小数点之前可有8位数字，小数点后有2位。

4、日期类型

Date 日期（日-月-年） DD-MM-YY(HH-MI-SS)

Timestamp 跟date比 它可以精确到微秒。精确范围0~9 默认为6.

## id、rowid、rownum的区别

rowid就是唯一标志记录物理位置的一个id。

而id是唯一标识记录的逻辑上的标识，所以rowid速度要快于id,是目前最快的 定位一条记录的方式

rowid和rownum都是"伪数列"

rownum用于标记结果集中结果顺序的一个字段，

这个字段被称为“伪数列”，也就是事实上不存在的一个数列。

它的特点是按顺序标记，而且是逐次递加的，

换句话说就是只有有rownum=1的记录，才可能有rownum=2的记录。

rownum关键字只能和<直接关联

如果是>或者=则需要给他起个别名

## 主键和唯一索引的区别？

在创建主键的同时会生成对应的唯一索引，主键在保证数据唯一性的同时不允许为 空，而唯一可以有一个为空数据项，一个表中只能有一个主键，但是一个主键可以 有多个字段，一个表中可以有多个唯一索引。

## Prepared statement和statement的区别

用Prepared statement进行开发。Prepared statement是预编译的，而statement不是，在每次执行sql语句的增删改时，如果是一条数据两者没差距，但如果数据量大于1，那么每次执行sql语句statement都要重新编译一次，而Prepared statement不用，Prepared statement的运行效率大于statement；从代码的可维护性和可读性来说，虽然用Prepared statement来代替statement会使代码多出几行，但这样的代码无论从可读性还是可维护性来说，都比直接使用statement的代码高很多档次；最重要的一点，从安全角度来说，使用Prepared statement可以打打提高程序的安全性，因为Prepared statement是用‘？’传参，具有安全性，而statement用的是‘+’字符串拼接，安全性较低。

## 数据库三范式

第一范式：数据库表中的所有字段值都是不可分解的原子值。

第二范式：需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部 分相关（主要针对联合主键而言）

第三范式：需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关

## 视图概述

视图可以视为“虚拟表”或“存储的查询”

创建视图所依据的表称为“基表”

视图的优点：

提供了另外一种级别的表安全性:隐藏了一些关键的字段

简化的用户的SQL命令

隔离基表结构的改变

## 存储过程概述

存储过程（Stored Procedure）

　　可以包含逻辑判断的sql语句集合。

　　是经过预编译，存在于数据库中。

　　通过调用指定存储过程的名字（可有参，可无参）来执行。

优点：

　　简化了复杂的业务逻辑，根据需要可重复使用

　　屏蔽了底层细节，不暴露表信息即可完成操作

　　降低网络的通信量，多条语句可以封装成一个存储过程来执行

　　设置访问权限来提高安全性

　　提高执行效率，因为它是预编译以及存储在数据库中

缺点：

　　可移植性差，相同的存储过程并不能跨多个数据库进行操作

　　大量使用存储过程后，首先会使服务器压力增大，而且维护难度逐渐增加

存储过程的语法：

--下面是在oracle数据库下最基本的语法

--仅创建一个名为testProcedure 的无参的存储过程

--IS也可以是AS

--如果已经存在名为 testProcedure 的存储过程，下面的语法会出现 名称已被使用的错误

--解决办法：

--第一句可以写成 create or replace procedure testProcedure

--这样会替换原有的存储过程

--NULL表示任何可以正确执行的sql 语句，但至少一句

create procedure testProcedure

IS

BEGIN

NULL

END;

存储过程的参数的分类:

IN

OUT

INOUT

注意：

　　存储过程之间可相互调用

　　存储过程一般修改后，立即生效。

## 索引概述

1、索引的概念

索引就是加快查询表中数据的方法。

数据库的索引类似于书籍的索引。

在书籍中，索引允许用户不必翻阅完整个书就能迅速地找到所需要的信息。

在数据库中，索引也允许数据库程序迅速地找到表中的数据，

而不必扫描整个数据库.

2、索引的特点

　　1.索引可以加快数据库的检索速度

　　2.索引降低了数据库插入、修改、删除等维护任务的速度

　　3.索引创建在表上，不能创建在视图上

3、索引的优点

　　1.创建唯一性索引，保证数据库表中每一行数据的唯一性

　　2.大大加快数据的检索速度，这也是创建索引的最主要的原因

　　3.减少磁盘IO（向字典一样可以直接定位）

4、索引的缺点

1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加

2.索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定 的物理空间

3.当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，降低 了数据的维护速度

5、索引的分类

　　1.普通索引和唯一性索引

　　 普通索引：CREATE INDEX mycolumn\_index ON mytable (myclumn)

　　 唯一性索引：保证在索引列中的全部数据是唯一的

　　 CREATE unique INDEX mycolumn\_index ON mytable (myclumn)

　　2. 单个索引和复合索引

　　 单个索引：即非复合索引

　　 复合索引：又叫组合索引，在索引建立语句中同时包含多个字段名，

　　 最多16个字段

　　 CREATE INDEX name\_index ON username(firstname,lastname)

　　顺序索引，散列索引,位图索引

## 必背的sql语句

一：oracle 分页

select \* from (select t.\*, rownum rn from (select \* from menu order by id desc) t where rownum < 10) where rn >=5

二: mysql 分页

select \* from music where id limit 5,5

三：oracle中如何快速将一张表的数据复制到另外一张表中(另外一张表不存在，另外一张 表存在，但数据为空)

1. .不存在另一张表时：

create table 新表 as select \* from 将要复制的表

1. 存在另一张表时：

insert into 新表名 select 字段 from 将要复制的表名

四：音乐专辑

查询出special <app:ds:special>表中的id 专辑名 并下面有多少首歌曲

Select s.id , min(s.sname),count(m.mid) from special s inner

join ms m on s.id=m.id group by s.id

五：快速删除一张表（不可事物回滚，也就是没有日志记录）

TRUNCATE from 表名

六：inner join

select 查找信息 from 表名 1 inner join 表名2 on 表名1.列名 = 表名2.列名

七：left join

左外连接 select 查找信息 from 表名1 left join 表名2 on 表名1.列名 = 表名2.列名

八：right join

右外连接 select 查找信息 from 表名1 right join 表名2 on 表名1.列名 = 表名2.列名

九：oracle中查询遍历树形结构（start with）

select \* from extmenu

start with pid=0

connect by prior id = pid

十：查询出来60-70,80-90,95-100学生的信息

select \* from stu where chengji between 60 and 70 or between 80 and 90 or between 95 and 100

select \* from stu where chengji > 60 and chengji < 70 or chengji > 80 and chengji < 90 or chengji > 95 and chengji < 100

十一：用exists替换in------进行联表查询

select \* from dept where exists(select \* from emp where emp.deptno=dept.deptno);

或

select \* from dept d inner join emp e on d.deptno = e.deptno(只查询出两表共同拥有的字段数据)

十二：删除表中的重复数据：

delete from xin a where a.rowid != (

select max(b.rowid) from xin b

where a.name = b.name

);

# 业务场景篇

## Spring的概述

Spring 是完全面向接口的设计，降低程序耦合性，主要是事务控制并创建bean实例对象。在ssh整合时，充当黏合剂的作用。IOC(Inversion of Control) 控制反转/依赖注入，又称DI(Dependency Injection) (依赖注入)

IOC的作用：产生对象实例，所以它是基于工厂设计模式的

Spring IOC的注入

　　 通过属性进行注入，通过构造函数进行注入，

　　 注入对象数组 注入List集合

　　 注入Map集合 注入Properties类型

Spring IOC 自动绑定模式：

可以设置autowire按以下方式进行绑定

按byType只要类型一致会自动寻找，

按byName自动按属性名称进行自动查找匹配.

AOP 面向方面（切面）编程

AOP是OOP的延续，是Aspect Oriented Programming的缩写，

　　　　意思是面向方面(切面)编程。

　　 注：OOP(Object-Oriented Programming ) 面向对象编程

AOP 主要应用于日志记录，性能统计，安全控制,事务处理（项目中使用的）等方面。

Spring中实现AOP技术：

在Spring中可以通过代理模式来实现AOP

代理模式分为

静态代理：一个接口，分别有一个真实实现和一个代理实现。

动态代理：通过代理类的代理，接口和实现类之间可以不直接发生联系，而 可以在运行期（Runtime）实现动态关联。

动态代理有两种实现方式，可以通过jdk的动态代理实现也可以通过cglib

来实现而AOP默认是通过jdk的动态代理来实现的。jdk的动态代理必须要有

接口的支持，而cglib不需要，它是基于类的。

Spring AOP事务的描述：

在spring-common.xml里通过<aop:config>里面先设定一个表达式，设定对service里那些方法 如：对add\* ,delete\*,update\*等开头的方法进行事务拦截。我们需要配置事务的传播（propagation="REQUIRED"）特性,通常把增,删,改以外的操作需要配置成只读事务（read-only="true"）.只读事务可以提高性能。之后引入tx:advice,在tx:advice引用 transactionManager（事务管理）,在事务管理里再引入sessionFactory,sessionFactory注入 dataSource，最后通过<aop:config> 引入txAdvice。

String实现ioc控制反转描述：

a对象想完成一个完整的功能，要先为自己的b类属性实例化，而在MVC模式中，这种现象很常见，为了简化这种实例化的工作，spring容器就产生了，它可以统一管理这种实例化频繁的操作，就是说这种本来应由自己实例化的工作交给Spring容器去控制了，也就是说控制反转了，实现的方案之一是在上述a类中提供一个关于b类的setter方法，这个方法会被Spring容器控制。

## 事务概述

在数据库中,所谓事务是指一组逻辑操作单元即一组sql语句。当这个单元中的一部分操作失败,整个事务回滚，只有全部正确才完成提交。

事务的ACID属性

1. 原子性（Atomicity）

原子性是指事务是一个不可分割的工作单位，事务中的操作要么都发生，

要么都不发生。

2. 一致性（Consistency）

事务必须使数据库从一个一致性状态变换到另外一个一致性状态。(数据不被破坏)

3. 隔离性（Isolation）

事务的隔离性是指一个事务的执行不能被其他事务干扰.

4. 持久性（Durability）

持久性是指一个事务一旦被提交，

它对数据库中数据的改变就是永久性的.

在JDBC中，

事务默认是自动提交的，

每次执行一个 SQL 语句时，如果执行成功，

就会向数据库自动提交，而不能回滚

为了让多个 SQL 语句作为一个事务执行：

（1）执行语句前调用 Connection 对象的 setAutoCommit(false);

以取消自动提交事务

（2）在所有的 SQL 语句都成功执行后，调用 commit(); 方法提交事务

（3）在出现异常时，调用 rollback(); 方法回滚事务。

## 权限概述

权限涉及到5张表：

用户表，角色表，权限表(菜单表)，用户角色关联表，角色权限关联表

当用户登录时，根据用户名和密码到用户表验证信息是否合法，如果合法

则获取用户信息，之后根据用户id再到用户角色关联表中得到相关连的角色

id集合,之后根据角色id再到角色权限关联表中获取该角色所拥有的权限id集合，

然后再根据权限id集合到权限表（菜单表）中获取具体的菜单，展现给当前

登录用户，从而达到不同用用户看到不同的菜单权限。

我们通过ZTree来给角色赋权并且通过ZTree来展示菜单，以及通过ZTree来管 理菜单即增加和编辑菜单。

我们做的权限控制到url级别,为了防止用户不登录直接输入url访问的这个弊端，通过拦截器进行拦截验证。

## OSCache业务场景

在我以前的项目中，我们考虑了系统性能问题，这个时候我们采用了Oscache缓存，刚开始把这个功能交给了项目组中的另外一个同事来做的，但是他做完的时候他发现缓存中明明已经缓存了数据，但是在取得时候发现没有数据，我们项目经理让我去帮忙看看这个问题，我阅读完他的代码之后，我发现了他每次缓存的时候都是调用一个新的缓存对象的方法，结果出现了明明已经走了缓存的方法而取不到数据的问题，通过我多年的工作经验，我就想到了应该用单例模式去封装一个单例工具类来调用oscache。但是，在后来的测试过程中，发现当并发访问的时候也会出现上述的问题，这个时候我直接采取的DCL（双重判定锁）单例模式封装了工具类，既解决了线程安全问题，相对的性能问题也考虑到了，这个问题才得到了完善的解决。

## 线程概述

线程的状态以及状态之间的相互转换：

　 1、新建状态(New)：新创建了一个线程对象。

　　2、就绪状态(Runnable)：线程对象创建后，其他线程调用了该对象的start()方法。该状态的线程位于可运行线程池中，变得可运行，等待获取CPU的使用权。

　　3、运行状态(Running)：就绪状态的线程获取了CPU，执行程序代码。

　　4、阻塞状态(Blocked)：阻塞状态是线程因为某种原因放弃CPU使用权，暂时停止运行。直到线程进入就绪状态，才有机会转到运行状态。阻塞的情况分三种：

　　(一)、等待阻塞：运行的线程执行wait()方法，JVM会把该线程放入等待池中。

　　(二)、同步阻塞：运行的线程在获取对象的同步锁时，若该同步锁被别的线程占用，则JVM会把该线程放入锁池中。

　　(三)、其他阻塞：运行的线程执行sleep()或join()方法，或者发出了I/O请求时，JVM会把该线程置为阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

　　5、死亡状态(Dead)：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

实现线程的两种方式：

是继承Thread类或实现Runnable接口，但不管怎样，当new了这个对象后，线程就已经进入了初始状态

wait和sleep的区别：

线程访问：

锁池状态，之后等待锁释放，然后访问代码

wait

等待队列(释放资源)--->调用notify或者notifyall之后锁池状态--->( 等待锁释放)--->可运行状态--->运行状态---->访问代码

sleep,join

不释放资源-->结束后直接进入可运行状态--->运行状态---->访问代码

一个java控制台程序，默认运行两个线程，一个主线程，一个垃圾回收线程。

线程与进程的区别：

1.线程(Thread)与进程(Process)

进程定义的是应用程序与应用程序之间的边界，通常来说一个进程就代表一个与之对应 的应用程序。不同的进程之间不能共享代码和数据空间，而同一进程的不同线程可以共 享代码和数据空间。

2.一个进程可以包括若干个线程，同时创建多个线程来完成某项任务，便是多线程。

## Ajax请求Session超时问题

我在做项目时有时会遇到session超时问题，如果session超时，平常请求没有什么问题，通过拦截器可以正确跳到登陆页面，可是你如果用ajax请求的话这就出现问题了，因为ajax是异步的，局部刷新，所以登陆界面不会再全页面中显示，他只会显示到页面的一部分当中。所以根据我这几年的经验找到了我认为比较好的一种方法。因为那我用的框架是和struts2集成的，所以就在拦截器中进行设置：

首先判断session是否为空就是判断session是否超时，如果超时就取出请求的head头信息request.getHeader("x-requested-with")，如果不为空就和XMLHttpRequest(Ajax标识)进行比较 (request.getHeader("x-requested-with").equalsIgnoreCase("XMLHttpRequest"))) 如果相等说明此请求是ajax请求。

如果是ajax请求就可以用response.setHeader("键","值")来设置一个标识来告诉用户这是ajax请求并且session超时时发出的，这样我就可以在回调函数中取出自己设置的那个唯一标识：XMLHttpRequest.getResponseHeader("");如果取出的值是和自己在后台中设置的值一样的话，就证明session已经超时，这样就可以设置window.location.replace("登陆界面")，来跳转到登陆界面了。

这样做虽然解决了问题，但是，会在每个回调函数中写入那些代码，这样的话代码就会显得特别零散，所以就想能不能定义一个全局的设置所以就找到了jqery的ajaxSetUp方法，通过ajaxSetUp对jqery的ajax进行全局的判断(ajaxSetUp就相当于ajax的拦截器)，通过设置ajaxSetUp里的complete，它就相当于回调函数，这样那就弥补了上一方法的不足。

我做项目时还用到$(document).ajaxStart()，这是ajax请求时的事件；$(document).ajaxSuccess()，这是AJAX请求成功后的事件。我一般用他们来显示遮罩层和隐藏遮罩层用的加遮罩层是为了不让用户重复提交，更提高了用户体验度，让用户知道已经提交了。

## 七：java线程池概述

java线程池的工作原理和数据库连接池的差不多，因为每次重新创建线程

都是很耗资源的操作，所以我们可以建立一个线程池，这样当需要用到线程

进行某些操作时，就可以直接去线程池里面找到空闲的线程，这样就可以直接

使用，而不用等到用到的时候再去创建，用完之后可以把该线程重新放入线程池

供其他请求使用从而提高应用程序的性能。

## OSCache概述

oscache是一个高性能的j2ee框架，可以和任何java代码进行集成，并且还可以通过标签对页面内容进行缓存，还可以缓存请求。

我们通常将那些频繁访问但是又不是经常改变的数据进行缓存。为了保证缓存数据的有效性，在数据发生改变的时候，我们要刷新缓存，避免脏数据的出现。刷新缓存的策略有两种，一种是定时刷新，一种手动刷新。

缓存数据的时机通常也分为两种，即在tomcat(web容器)启动时候加载数据进行缓存，另外也可以在用户第一次访问数据的时候进行缓存，这个相当于缓存的立即加载和按需加载。

缓存的层次如下:jsp-->action-->service-->dao,缓存越靠前对性能的提升越大

一个action里面可以有多个service,一个service中可以有多个dao或者多个service

任何类之间都可以进行相互调用，可以通过构造函数传参，set,get传参或者是方法传 参将相关的类连接起来。

## OSCache+autocomplete+单例业务场景

在我以前做某项目的过程中，其中我们在做产品列表的查询的时候为了提高用户的体验度，我们使用了autocomplete插件来代替select进行品牌的选择，才开始的时候每次都要根据用户输入的信息去查询数据库进行模糊匹配返回结果，后来我们考虑到系统的性能，因此我们采用了oscache缓存，才开始这个功能是交给我们项目组中的另外一个同事来做的，但是他做完后，我们在使用这个工具类的时候，发现有时缓存中明明已经有时我们需要的数据，但是从缓存里面取的时候，发现没有，之后项目经理让我去帮这个同事看看这个问题，我经过阅读他的代码发现，它里面在使用缓存的时候，针对于每次方法的调用都产生一个新的实例，结果导致了上面的问题，这个时候我想起了可以使用设计模式中的单例模式来解决这个问题，才开始我直接采用了普通的单列模式，但是后来在测试的过程中，发现当用户并发量大的时候还是会出现上面的问题，之后我再次考虑了代码，最后发现是因为没有给单列模式加锁的原因，从而导致了大用户并发的时候，线程安全的问题，之后我便在方法上加上了synchronized关键字，解决上述的问题，但是后来测试人员反馈，觉的这段的性能有问题，我考虑之后便采用在方法体内加锁并结合双重判定的方式解决了上面的问题。我们是将数据在tomcat启动的时候加载到缓存中，之后用户进行查询的时候直接从缓存中获取数据，根据前缀匹配进行查询，将结果返回给用户。这样在提高用户体验度的同时也提高性能。

## 缓存概述

应用程序为了提高性能，可以通过使用缓存来达到目的，缓存的存储介质可以内存或者硬盘，通常将数据存储在内存里，确切的说是jvm的内存中，缓存是基于Map这种思想构建的，以键值对的方式进行存储和获取，之所以还可以将缓存的数据存储在硬盘中，是因为内存资源相当有限和宝贵，所以当内存资源不足的时候，就可以将其存储到硬盘中，虽然硬盘的存取速度比内存要慢，但是因为减少了网络通信量，所以还是提高程序的性能。缓存可以分为客户端缓存和服务器端缓存，所谓的客户端缓存通常指的是IE浏览器的缓存，服务器端缓存指的web服务器的缓存，通常可以通过第三方组件实现，如oscache,memcache。

## 实现页面静态化业务场景

我们在做某项目时，涉及到程序访问的性能问题，这时候我们想到可以通过静态化来提高用户访问时候的性能，所以我们就采用了freemarker模板引擎，考虑到页面也是要有动态的变化的，所以我们采用spring定时器在每天晚上2点钟的时候定时再次生成html静态页面，考虑发布时候的性能问题，我们又采取线程池技术，让多个线程同时发布，从而缩减发布时间。

## servlet线程安全描述

servlet是单列的，对于所有请求都使用一个实例，所以如果有全局变量被多

线程使用的时候，就会出现线程安全问题。

解决这个问题有三种方案:

1.实现singleThreadModel接口，这样对于每次请求都会创建一个新的servlet实例，这样就会消耗服务端内存，降低性能，但是这个接口已经过时，不推荐使用。

2.可以通过加锁(synchroniezd关键字)来避免线程安全问题。这个时候虽然还是单列，但是对于多线程的访问，每次只能有一个请求进行方法体内执行，只有执行完毕后，其他线程才允许访问，降低吞吐量。

3.避免使用全局变量，使用局部变量可以避免线程安全问题，强烈推荐使用此方法来解决servlet线程安全的问题。

## (jbpm4)工作流引擎描述:

JPBM是JBOSS旗下的一个开源的基于hibernate的工作流引擎。工作流就是在日常生活中，我们一些常见的如请假流程、采购流程、入职流程，通俗的来讲就是一些在现实生活中的流程以信息化以程序的方式实现。

一个工作流首先需要进行流程定义，流程定义是由节点和跳转组成的，节点又可以称为环节、活动节点、活动环节，并且节点也可以分为两大类型：人工节点和自动节点，人工节点有start开始节点、end结束节点、task任务节点，自动节点有decision判断节点、fork分支节点、join聚合节点和state状态节点，并且一个流程有且只有一个开始节点，但可以有多个结束节点。

流程定义是静止的，它在运行状态时会转换成流程实例，一个流程定义可以对应多个流程实例。流程运行后，会产生两个文件，\*.jdpl.xml文件和\*.png图片文件，也会生成18张数据库表，常用且核心的表有JBPM4\_LOB 存储表，主要存储xml文件和png图片、JBPM4\_TASK 任务表、JBPM4\_EXECUTION 流程实例表、JBPM4\_VARIABLE变量表。

JBPM有五大核心类：

ProcessEngine：主要获取各种的Service

RepositoryService：主要发布流程定义

ExecutionService：主要操作流程实例

TaskService：主要操作人工服务

HistoryService：主要操作历史服务。

核心方法：

读取jbpm定义的文件生成zip包存到lob表中：createDeployment()

获取流程定义列表：createProcessDefinitionQuery

根据定义的key或id来启动流程实例：startProcessInstanceByKey(id)

获取待办任务列表：findPersonalTasks(userName)

完成指定任务列表：completeTask(\*.getActivityId())

获取历史任务列表：createHistoryTaskQuery()

获取流程实例的ID：task.getExecutionId()

(了解的表)

JBPM4\_HIST\_ACTINST 流程活动(节点) 实例表

JBPM4\_HIST\_DETAIL 流程历史详细表

JBPM4\_HIST\_PROCINST 流程实例历史表

JBPM4\_HIST\_TASK 流程任务实例历史表

JBPM4\_HIST\_VAR 流程变量( 上下文) 历史表

## JBPM业务场景

首先进行请假的流程定义，我们流程的定义是（员工提交请假单---》经理审批---》总监审批---》总经理审批---》结束），通过repositoryService将其发布部署到jbpm4\_lob表中，

之后获取流程定义列表，选中请假的流程定义，员工开始进行请假单的填写，保存并通过executionService开启流程实例，然后用taskService获取经理的待办任务列表，选中待办任务，进行审批，通过调用taskService.completeTask()进入到总监审批环节，然后用总监进行登录，同样获取待办任务列表，然后调用taskService.completeTask()进入总经理审批环节，总经理审批之后，结束流程。在这个过程中我们还可以根据historyService查看当前登录人已办的任务列表。

## Ant描述

Ant是apache下的对项目进行自动打包、编译、部署的构建工具，他主要具有 轻量级并且跨平台的特性，而且基于jvm，默认文件名为build.xml

Ant主要的标签：

Project 根标签，

target 任务标签，

property 属性标签，自定义键/值 供多次使用，

java 执行编译后的java文件

javac 编译java文件

war 打成war包

其它标签：copy，delete，mkdir，move，echo等。

## FreeMarker描述

FreeMarker是一个用Java语言编写的模板引擎，它是基于模板来生成文本输出的通用工具。Freemarker可以生成HTML， XML，JSP或Java等多种文本输出。

工作原理：定义模板文件，嵌入数据源，通过模板显示准备的数据

（数据 + 模板 = 输出）

　　我们在使用模板中发现freemarker具有许多优点，它彻底的分离表现层和业务逻辑，模板只负责数据在页面中的表现，不涉及任何的逻辑代码，所以使得开发过程中的人员分工更加明确，作为界面开发人员，只需专心创建HTML文件、图像以及Web页面的其他可视化方面，不用理会数据；而程序开发人员则专注于系统实现，负责为页面准备要显示的数据。

如果使用jsp来展示，开发阶段进行功能调适时，需要频繁的修改JSP，每次修改都要编译和转换，浪费了大量时间，FreeMarker模板技术不存在编译和转换的问题，在开发过程中，我们在不必在等待界面设计开发人员完成页面原型后再来开发程序。由此使用freemarker还可以大大提高开发效率。

## webService描述

webservice是SOA（面向服务编程）的一种实现，主要是用来实现异构平台通信也就 是不同平台不同项目之间的数据传输，从而避免了信息孤岛的问题，它之所以能够

进行异构平台通信是因为它是完全基于xml的，所以说，webService是跨平台，

跨语言，跨框架的。

webservice的三要素分别是：

wsdl（webservice description language）用来描述发布的接口（服务）

soap(simple object access protocol) 是xml和http的结合，是webservice数据通信

的协议

uddi（univers）用来管理,分发,查询webService的服务

webservice的具体三种实现方式（框架）

1. Axis2:可以用多种语言开发（java c c++ C#），是一个重量级框架，功能非常 强 大，所以说涉及到的api非常多，所以说它的性能比较慢。

2. Xfire：它相比Axis2来说是一个轻量级框架，但是它的性能要比Axis2高， Api也相对少。

3. cxf：是Xfire的升级版，就好比是，struts2是webwork的升级，人们会逐渐淡忘webwork,但是struts2会慢慢变成主流。然而cxf就好比是struts2，会成为主流技术。然后cxf和spring集成起来非常方便，简易，性能方面也要比Xfire高。

【**注**】jdk6 自带的webservice jws

**业务场景**

我在以前做项目的时候，其中遇到一个功能，需要进行两个项目之间的数据的传输，

项目经理让我去完成这个任务，我根据以往的项目经验，想到两种解决方案，第一种

就是开放另外一个项目的数据库的权限给我，然后我直接通过访问另外一个项目的数据

库，来得到需要的信息，但后来我分析了下，觉的这种方式不安全，而且因为当时

这个项目是另外一家公司负责在做，所以数据库里面的表结构，以及以后牵涉

到的责任问题都很多，所以我就采用了第二种方案，即通过webservices的方式，进行

异构系统之间数据信息的传递，webservices的具体实现，有xfire,cxf,axis2,

我根据以往的项目经验，了解到cxf是xfire的升级版本，适用于java语言，

xfire/cxf 性能比axis2要高，并且和spring整合起来也比较方便，而axis2支持更多的 语言，性能相对于cxf要低，通过上面分析，结合我们目前的两个项目都是基于java 语言的，所以我采用cxf这种方式实现了两个项目之间数据的传递，我们为了保证 webservice的安全性我们采用了基于WS-Security标准的安全验证(使用CXF回调函数)。

**webservice服务端配置流程**

首先在web.xml中引入cxfServlet核心类，指定对以/cxf开头的url路径提供webservice服务，之后我们在要发布成webservice接口上添加@Webservice 注解，而且还要在实现类上添加同样的webservice注解并且要说明实现了哪个接口，之后在spring-webservice.xml中发布webservice服务，通过jaxws:endpoint这个标签，并且在标签配置implementor和address来表明实现服务的类，以及发布的地址，最后在浏览器中输入相关的webservice地址?wsdl来验证服务是否发布成功。

**webservice客户端的配置**

首先通过wsdl2java根据发布的webservice服务端地址的wsdl生成客户端调用的中间桥梁java类，将生成的java类拷贝到客户端项目中，配置spring-client.xml文件，通过jaxws:client定义一个bean,并通过address属性指明要访问的webservice的服务地址，通过serviceClass指明充当中间桥梁的服务类，之后获取该bean,就可以通过它来访问发布的webservice接口中的方法。

## oracle索引概述

索引呢是与表相关的一个可选结构，可以提高sql语句的检索效率，相当于我们的字典目录 ，可以快速进行定位 ，所以可以减少磁盘I/O, 但是因为索引在物理与逻辑上都是独立于表的数据 它会占用一定的物理空间(额外磁盘空间) 所以并不是索引越多越好,而我们应该根据业务需求去创建索引,而且进行增删改操作时 oracle又要自动维护索引 所以在一定程度上也降低了维护速度，而且我们在创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加，我们一般创建索引呢 是这样创建的 create index 索引名 on 表名(字段)，索引又分为普通索引 唯一索引(unique) 单个索引 复合索引(又叫组合索引，在索引建立语句中同时可包含多个字段名)，顺序索引，散列索引,位图索引。

## oracle存储过程

存储过程就是封装一些sql的集合，也就是一条条的sql语句，过程的优点就是简化了sql命令加上它是预编译的，所以它的执行效率和性能较高，再者，如果不调用过程的话就要和数据库发生多次交互，调用过程只需传一个命令所有的那些执行逻辑都在数据库端执行，所以说它降低了网络的通信量，其次，存储过程大大提高了安全性，这就是优点

缺点呢，就是不同的数据库对过程支持的关键字支持的关键字都是不一样的，所以它的移植性是非常差的，再者，它的维护性难度也比较大，因为它没有专业的调试和维护工具，所以说它维护起来比较麻烦，这就是存储过程的基本概述

## 二十、Junit 业务场景

在我们开发项目的时候为了提高代码的性能和保证逻辑正确性，在我们编写代码后往往都要进行单元测试，来验证代码，当时我们公司开发人员全部使用的main方法来进行验证，但是使用mian的最大缺点就是不能将多个类同时进行验证，验证的结果不直观，测试复杂（每个类都要写main方法，单个运行），一定程度上浪费时间，所有我和项目经理提议使用专业测试工具Junit来进行测试，因为Junit是一个Java语言的单元测试框架 ，测试简单，不仅可以提供工作效率和代码的质量，也提高团队的合作能力，我提议后我们进行了Junit的培训使用Junit4加注解的方式来测试。

## 二十一、Apache+Tomcat 实现负载均衡及seesion复制

当我们tomcat访问量大,线程连接数不够时,我们考虑到了tomcat的负载均衡来分担过多的访问.性能方面负载均衡也能利用多台tomcat来增大内存量,

**流程**,准备工作apache,Jk\_mod,tomcat,在apache的conf/httpd.conf文件中 使用include 标签引入我们自定义的一个mood\_jl.conf,在modules中引入下载的k\_mod-apache-X.X.XX.so文件,在其中引入我们的.so,及work.properties文件,及指定负载分配控制器controller,在work.properties文件中worker.list=controller,tomcat1,tomcat2指定service,worker.tomcat1.port Ajp端口号,type 是ajp,host为指定ip,lbfactor 指定分配权重值越大分担请求越多,worker.controller.type=lbworker.controller.balanced\_workers=tomcat1,tomcat2 指定分担请求的tomcat Session的复制在tomcat中service.xml中Engine标签加入 jvmRoute 值为work,properties中指定的tomcat名称,然后打开<Cluster标签的注释,最后在应用中程序的web.xml文件中增加<distributable/>。

## 二十二、Ant业务场景

Ant是基于java语言编写的，因此具有跨平台的特性，此外还具有简洁方便，灵活

配置的特性，因此我就在XX项目中使用ant进行项目的编译，打包，部署操作。使用ant

之后，如果我们在客户那里修改代码后，就可以直接使用ant进行编译，打包，部署，而不需要为了编译，打包，部署专门在客户那里安装eclipse.此外使用ant也可以直接和svn进行交互，下载源码的同时进行编译，打包，部署。

## 二十三、maven业务场景

前段时间在研究maven，知道maven是一个项目管理工具，其核心特点就是通过

maven可以进行包的依赖管理，保证jar包版本的一致性，以及可以使多个项目共享

jar包，从而能够在开发大型j2ee应用的时候，减小项目的大小，并且和ant

比起来，maven根据“约定优于配置”的特性，可以对其项目的编译打包部署进行了

更为抽象的封装，使得自己不需要像ant那样进行详细配置文件的编写，直接使用

系统预定好的mvn clean,compile,test,package等命令进行项目的操作。于是我就

在XX项目中采用了maven,为了保证团队中的成员能够节省下载jar包所需要的时间，

于是我就采用nexus搭建了在局域网内的maven私服，然后通过配置settings.xml中

建立mirror镜像，将所有下载jar包的请求都转发到maven私服上，之后通过在pom.xml

即(project object model)中配置项目所依赖的jar包，从而达到在构建项目的时候，

先从本地仓库中查找，如果不存在从内部私服查找，如果不存在最后再从外网central

服务器查找的机制，达到了节省下载带宽，提高开发效率，以及jar包重用的目的。

## 二十四、Servlet的概述：

Servlet是一个web容器，我们通常用的servlet是httpservlet，而httpservlet又是继承于genericservlet，而genericservlet又实现了servlet接口

servlet的生命周期是 ：先进行实例化，然后是初始化，然后是提供服务，然后销毁，最后不可用，在这五个生命周期，其中，初始化是调用的init方法，这个方法只有一个，而提供服务的时候调用的是service方法，而我们具体在我们所写的这个方法中，因为我们继承了httpservlet，其实就是对应了doGet（），doPost(),这种方法，然后据我了解，servlet是单例的。非线程安全的，我们通常有一下几种方案来解决：

第一种，继承SingleThreadModel但是这样每次都会创建一个新的servlet实例，但这样消耗服务器的内存，降低了性能，并且这个接口现在已经过时了，不推荐使用。

第二种：我们尽量避免使用全局变量，就我个人而言，我比较喜欢使用这种方法。

第三种，我们可以通过使用ThreadLocal， 内部结构是一个Map结构，用当前线程作为key,他会创建多个副本。get,set方法

第四种，我们当然还可以来加锁，进行解决线程问题。

而且我还知道，向我们这种常用的MVC框架，struts1，spring这些MVC框架，都是基于servlet发展而来的，就比如struts1 的核心总控制器是ActionServlet，而springMVC的前端总控制器是dispatchServlet，在项目我们曾经用serlet来生成 图片验证码的，防止用户进行暴力破解

（别人问了，再回答）

servlet的配置文件 web.xml

<servlet>

<servlet-name>ImageCodeServlet</servlet-name> <servlet-class>org.leopard.code.ImageCodeServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>ImageCodeServlet</servlet-name>

<url-pattern>/d</url-pattern>

</servlet-mapping>

描述：

我在web.xml中，我首先需要写一个servlet标签，servlet标签中有两个子标签，一个叫servlet-name，这个name可以随便起，但是要保证唯一性，除此之外，在这个servlet-name下有一个servlet-class，这个servlet-class对应的就是我后台提高服务的servlet，除此之外还有一个servlet-mapping，这个里边首先有一个servl-name。，这个servl-name首先要保证和上边的servlet-name保持一致，除此之外还有一个url-pattern，这是一个虚拟路径，是用来发送请求的url地址

# 优化篇

## 代码优化

代码结构层次的优化(目的:更加方便代码的维护--可维护性，可读性)

1.代码注释(代码规范)

2.工具类的封装(方便代码的维护，使代码结构更加清晰不臃肿，保证团队里代码 质量一致性)

3.公共部分的提取

代码性能的优化(目的:使程序的性能最优化)

1.使用一些性能比较高的类(bufferInputStream)

2.缓冲区块的大小(4k或者8k)

3.公共部分的提取

4.通常要用stringbuffer替代string加号拼接

## 业务优化

我们做项目的时候业务优化这方面最主要是从用户体验度角度进行考虑,减少用户操 作的步骤提高工作效率，通常有以下几种：

　　1.可以通过tabindex属性来改变tab键盘的操作顺序

　　2.可以通过回车键来进行搜索或者提交操作

　　3.对于单选按钮和复选按钮可以通过操作后面的文本来选择前面的单选按钮以及复选 按钮

　　4.添加的信息要按照id倒序进行排列

　　5.进行搜索操作时加入js loading操作（不仅告诉用户所进行的请求正在被处理，而 且防止用户多次点击提交操作）

　　6.当进行删除操作的时候要弹出提示框，警告用户要进行删除操作，是否确认。

　　7.根据returnURL在用户登录成功后直接跳到想要访问的资源。

　　8.进行删除操作时通过confirm提示用户是否确认删除操作，操作完后提示操作是否 成功。

　　9.减少用户操作的步骤

10.使用autocomplete插件快速进行搜索

## sql优化

1、SELECT子句中避免使用 \*， 尽量应该根据业务需求按字段进行查询

2、尽量多使用COMMIT如对大数据量的分段批量提交释放了资源，减轻了服务器压力

3、在写sql语句的话，尽量保持每次查询的sql语句字段用大写，因为oracle总是先解析 sql语句，把小写的字母转换成大写的再执行

4、用UNION-ALL 替换UNION，因为UNION-ALL不会过滤重复数据，所执行效率 要快于UNION,并且UNION可以自动排序，而UNION-ALL不会

5、避免在索引列上使用计算和函数,这样索引就不能使用

## 防sql注入

针对防sql注入，我们通常是这样做的：

首先在前台页面对用户输入信息进行js验证，对一些特殊字符进行屏蔽，

比如：or ,单引号，--，= ，还有就是限制用户名输入的长度，我们一般

将其限制在6---13位。另外，对于用户的敏感信息我们进行Md5加密，还有

，为了增加用户体验度和用户友好度，为了不使用户看到一些详细的异常信息

我们会进行错误信息页面的定制，像404,500错误。另一个我层面讲，这样做

也是为了保护我们的一些重要信息。此外，我们会给特定的人分配定定的权限

，而不是给其分配管理员权限！