1.讲解一下Java跨平台原理

由于操作系统的的指令集不是完全一致的，就会让我们的程序在不同的操作系统上执行不同的程序代码

Java通过不同的系统，不同版本不同位数的java虚拟机来屏蔽不同的系统指令集差异而对外提供统一的接口（Java API）。我们只需按照接口开发即可。

2.面向对象的特征

封装：把客观的事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或对象进行操作，对不可信的进行信息隐藏。

抽象：就是把现实生活中的对象抽象为类

继承：子类继承父类的特征和行为，使得子类具有父类的各种属性和方法。或子类从父类继承方法，使得子类具有父类相同的行为。

多态：多态的特征是表现出多种形态，具有多种实现方式。或者多态是具有表现多种形态的能力的特征。或者同一个实现接口，使用不同的实例而执行不同的操作。

3.装箱与拆箱

Java有4类8中基本数据类型同时还有他们对应的包装类型例如int的包装类型是Integer

自动装箱

Integer i = 1;//自动装箱

1

实际上在编译时会调用Integer.valueOf(i)方法来装箱。

static final int low = -128;

static final int high = 127;

1

2

@HotSpotIntrinsicCandidate

public static Integer valueOf(int i) {

if (i >= IntegerCache.low && i <= IntegerCache.high)

return IntegerCache.cache[i + (-IntegerCache.low)];

return new Integer(i);

}

1

2

3

4

5

6

7

自动拆箱

int j = i; //自动拆箱

1

同样在编译时会调用intValue()方法

@HotSpotIntrinsicCandidate

public int intValue() {

return value;

}

1

2

3

4

缓存值

我们会发现在valueOf中，当 i 的值在IntegerCache.low 和 IntegerCache.high 之间时会将i的值放入一个缓冲数组里，所以当我们再次创建相同对象时将直接从缓冲数组中进行提取，即指向同一个对象。

private static class IntegerCache {

static final int low = -128;

static final int high;

static final Integer cache[];

static {

// high value may be configured by property

int h = 127;

String integerCacheHighPropValue =

VM.getSavedProperty("java.lang.Integer.IntegerCache.high");

if (integerCacheHighPropValue != null) {

try {

int i = parseInt(integerCacheHighPropValue);

i = Math.max(i, 127);

// Maximum array size is Integer.MAX\_VALUE

h = Math.min(i, Integer.MAX\_VALUE - (-low) -1);

} catch( NumberFormatException nfe) {

// If the property cannot be parsed into an int, ignore it.

}

}

high = h;

cache = new Integer[(high - low) + 1];

int j = low;

for(int k = 0; k < cache.length; k++)

cache[k] = new Integer(j++);

// range [-128, 127] must be interned (JLS7 5.1.7)

assert IntegerCache.high >= 127;

}

private IntegerCache() {}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

Question：有了基本数据类型为什么还要包装类型

Answer：我们知道Java是一个面相对象的编程语言，基本类型并不具有对象的性质，为了让基本类型也具有对象的特征，就出现了包装类型（如我们在使用集合类型Collection时就一定要使用包装类型而非基本类型），它相当于将基本类型“包装起来”，使得它具有了对象的性质，并且为其添加了属性和方法，丰富了基本类型的操作。

另外，当需要往ArrayList，HashMap中放东西时，像int，double这种基本类型是放不进去的，因为容器都是装object的，这是就需要这些基本类型的包装器类了。

4.==与equals的区别

==用来判断两个变量之间的值是否相等。变量可以分为基本数据类型和引用类型，如果是基本数据类型的变量直接比较值，而引用数据类型要比较对应的应用的内存的首地址。

equals用来比较两个对象长得是否一样，判断两个对象的某些特征是否一样。因此需要重写equals方法。

5.String、StringBuffer、StringBuilder

String是内容不可变的字符串，而StringBuffer和StringBuilder则是内容可以改变的字符串。String底层使用了一个不可变的字符数组（用final修饰），StringBuffer和StringBuilder底层使用的可变的字符数组（没有使用final来修饰）。

public final class String

implements java.io.Serializable, Comparable<String>, CharSequence {

@Stable

private final byte[] value;

1

2

3

4

5

6

abstract class AbstractStringBuilder implements Appendable, CharSequence {

/\*\*

\* The value is used for character storage.

\*/

byte[] value;

1

2

3

4

5

6

StringBuffer和StringBuilder都继承自AbstractStringBuilder，StringBuilder是线程不安全的，效率较高，而StringBuffer是线程是线程安全的，效率较低。

字符串的拼接

String str="abc";

System.out.println(str);

str=str+"de";

System.out.println(str);

1

2

3

4

5

6

7

String str="abc"+"de";

StringBuilder stringBuilder=new StringBuilder().append("abc").append("de");

System.out.println(str);

System.out.println(stringBuilder.toString());

1

2

3

4

5

6

7

这样输出结果也是“abcde”和“abcde”，但是String的速度却比StringBuilder的反应速度要快很多，这是因为第1行中的操作和

String str=”abcde”;

是完全一样的，所以会很快，而如果写成下面这种形式

String str1=”abc”;

String str2=”de”;

String str=str1+str2;

那么JVM就会不断的创建、回收对象来进行这个操作了。速度就会很慢。

String：适用于少量的字符串操作的情况

StringBuilder：适用于单线程下在字符缓冲区进行大量操作的情况

StringBuffer：适用多线程下在字符缓冲区进行大量操作的情况

6.Java中的集合

Java中的集合分为value，key-value（Collection，Map）两种。

存储值又分为List和Set：List 有序可重复。Set 无序不可重复，根据equals和hashCode方法来判断，也就是说如果一个对象存储在Set中，必须重写equals和hashCode方法。

ArrayList和LinkedList的区别：

ArrayList底层使用的是数组，LinkedList底层使用的是链表。

transient Object[] elementData; // non-private to simplify nested class access

1

2

transient int size = 0;

/\*\*

\* Pointer to first node.

\*/

transient Node<E> first;

/\*\*

\* Pointer to last node.

\*/

transient Node<E> last;

...........

private static class Node<E> {

E item;

Node<E> next;

Node<E> prev;

Node(Node<E> prev, E element, Node<E> next) {

this.item = element;

this.next = next;

this.prev = prev;

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

数组查询元素比较快，而插入、删除和修改比较慢（数组在内存中是一块连续的内存，如果删除或插入时需要移动内存）

链表不要求内存是连续的，在当前元素中存放上一个或下一个元素的地址。查询需要从头部开始一个一个的找，所以查询效率低，插入时不需要移动内存，只需改变引用指向即可，所以插入或者删除效率较高。

7.HashMap和HashTable的区别

HashMap和HashTable都可以用来修饰key-value的数据。

HashMap是可以把null作为key或者value的，而HashTable是不可以的。

HashMap是线程不安全的，效率较高，而HashTable是线程安全的，效率低。

Question：即想线程安全又想效率高？

Answer： ConcurrentHashMap，通过把整个Map分为N个Segment（类似HashTable）可以提供相同的线程安全同时可以将效率提升N倍，默认提升16倍。

8.实现一个拷贝文件的工具类使用字符流还是字节流？

Answer：我们拷贝的文件不确定是只包含字符流，有可能有字节流（图片、声音、图像等），为考虑适用性，要使用字节流。

9.线程

Question1：线程有几种实现方式？

Answer1： 2种。继承Thread类实现一个线程或通过实现Runnable接口。Thread类本质上也是实现Runnable接口。

public class Thread implements Runnable {

1

Question2：如何启动？

Answer：启动线程使用start()方法，而启动以后执行的是run()方法。

public class MyThread implements Runnable {

@Override

public void run() {

// TODO Auto-generated method stub

}

}

1

2

3

4

5

6

7

8

9

10

11

Thread thread = new Thread(new MyThread());

thread.start();

1

2

Question3:怎么区分线程？在一个系统中有很多线程，每个线程都会打印日志，我想区分是哪个线程打印的怎么办？

Answer： thread.setName(“线程名字”);这是一种规范，在创建线程后需要设置线程名称。

10.线程并发库

Java通过Exectors提供四个静态方法创建四种线程池：

方法名 作用

newFixedThreadPool(int nThreads) 创建固定数量的线程池

newCachedThreadPool() 创建缓存的线程池

newSingleThreadExecutor() 创建单个线程

newScheduledThreadPool(int corePoolSize) 创建定时器线程池

1、提高资源利用率

线程池可以重复利用已经创建了的线程

2、提高响应速度

因为当线程池中的线程没有超过线程池的最大上限时，有的线程处于等待分配任务状态，当任务到来时，无需创建线程就能被执行。

3、具有可管理性

线程池会根据当前系统特点对池内的线程进行优化处理，减少创建和销毁线程带来的系统开销。

11.设计模式

设计模式就是经过前任无数次的实践总结的，设计过程中可以反复使用的，可以解决特定的问题的设计方法。

常用的设计模式：

单例模式：是一种常用的软件设计模式，在它的核心结构中值包含一个被称为单例的特殊类。一个类只有一个实例，即一个类只有一个对象实例。

单例模式可以分为懒汉式和饿汉式：

懒汉式单例模式：在类加载时不初始化。

饿汉式单例模式：在类加载时就完成了初始化，所以类加载比较慢，但获取对象的速度快。

第一种（懒汉，线程不安全）:

public class SingletonDemo1 {

private static SingletonDemo1 instance;

private SingletonDemo1(){}

public static SingletonDemo1 getInstance(){

if (instance == null) {

instance = new SingletonDemo1();

}

return instance;

}

}

1

2

3

4

5

6

7

8

9

10

这种写法lazy loading很明显，但是致命的是在多线程不能正常工作。

第二种（饿汉）：

public class Singleton {

private static Singleton instance = new Singleton();

private Singleton (){}

public static Singleton getInstance() {

return instance;

}

}

1

2

3

4

5

6

7

工厂模式：工厂模式（Factory Pattern）是 Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

步骤 1

创建一个接口:

Shape.java

public interface Shape {

void draw();

}

1

2

3

步骤 2

创建实现接口的实体类。

Rectangle.java

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Inside Rectangle::draw() method.");

}

}

1

2

3

4

5

6

7

8

Square.java

1

public class Square implements Shape {

@Override

public void draw() {

System.out.println("Inside Square::draw() method.");

}

}

1

2

3

4

5

6

7

Circle.java

1

public class Circle implements Shape {

@Override

public void draw() {

System.out.println("Inside Circle::draw() method.");

}

}

1

2

3

4

5

6

7

8

步骤 3

创建一个工厂，生成基于给定信息的实体类的对象。

ShapeFactory.java

public class ShapeFactory {

//使用 getShape 方法获取形状类型的对象

public Shape getShape(String shapeType){

if(shapeType == null){

return null;

}

if(shapeType.equalsIgnoreCase("CIRCLE")){

return new Circle();

} else if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new Rectangle();

} else if(shapeType.equalsIgnoreCase("SQUARE")){

return new Square();

}

return null;

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

步骤 4

使用该工厂，通过传递类型信息来获取实体类的对象。

FactoryPatternDemo.java

public class FactoryPatternDemo {

public static void main(String[] args) {

ShapeFactory shapeFactory = new ShapeFactory();

//获取 Circle 的对象，并调用它的 draw 方法

Shape shape1 = shapeFactory.getShape("CIRCLE");

//调用 Circle 的 draw 方法

shape1.draw();

//获取 Rectangle 的对象，并调用它的 draw 方法

Shape shape2 = shapeFactory.getShape("RECTANGLE");

//调用 Rectangle 的 draw 方法

shape2.draw();

//获取 Square 的对象，并调用它的 draw 方法

Shape shape3 = shapeFactory.getShape("SQUARE");

//调用 Square 的 draw 方法

shape3.draw();

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

步骤 5

执行程序，输出结果：

Inside Circle::draw() method.

Inside Rectangle::draw() method.

Inside Square::draw() method.

1

2

3

代理模式：在代理模式（Proxy Pattern）中，一个类代表另一个类的功能。这种类型的设计模式属于结构型模式。

在代理模式中，我们创建具有现有对象的对象，以便向外界提供功能接口。

装饰器模式：装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。

这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法签名完整性的前提下，提供了额外的功能。

12.http Get和Post请求区别

Get和Post请求都是http的请求方式，用户通过不同的http的请求方式完成对资源（url）的操作。Get、Post、Put、Delete就对应着这个资源的查、改、增、删4个操作，具体讲Get请求一般用于获取/查询资源信息，而Post一般用于更新资源信息。

区别：

1.Get请求提交的数据全在地址栏显示出来，而Post请求不在地址栏显示出来。

2.传输数据的大小，Get由于地址栏的长度限制。

3。安全性，Post的安全性要比Get的安全性高。

13.Servlet

servlet全称Java servlet ，是用Java编写的服务器端程序，servlet是指任何实现了servlet接口的类。其主要功能在于交互式地浏览和修改数据，生成动态的web。

我们通常通过继承HttpServlet重写doGet和doPost方法或重写service方法完成对get和post请求的响应。

servlet的生命周期：

加载servlet的class —>实例化servlet —>调用servlet的init方法完成初始化 —>响应请求（servlet的service方法）—>servlet容器关闭（servlet的destory方法）

servlet API中的forward与redirectde区别：

1.forward是服务器的转向而redirect是客户端的跳转。

2.使用forward浏览器的地址不会发生改变，而direct会发生改变。

3.forward是在一次请求中完成，而redirect是重新发起请求。

4.forward是在服务器端完成，而不用客户端重新发起请求，效率较高。

14.Jsp和servlet异同

jsp是servlet技术的扩展，所有的jsp文件都会被翻译为一个继承HttpServlet的类，也就是jsp最终也是一个servlet，这个servlet对外提供服务。

servlet和jsp最主要的不同点在于，servlet如果要实现html的功能，必须使用write输出对应的html，而jsp是由html和Java组成便于显示视图。

jsp侧重于视图

servlet主要用于控制逻辑

15.jsp的九大内置对象

1、request对象

request 对象是 javax.servlet.httpServletRequest类型的对象。 该对象代表了客户端的请求信息，主要用于接受通过HTTP协议传送到服务器的数据。（包括头信息、系统信息、请求方式以及请求参数等）。request对象的作用域为一次请求。

2、response对象

response 代表的是对客户端的响应，主要是将JSP容器处理过的对象传回到客户端。response对象也具有作用域，它只在JSP页面内有效。

3、session对象

session 对象是由服务器自动创建的与用户请求相关的对象。服务器为每个用户都生成一个session对象，用于保存该用户的信息，跟踪用户的操作状态。session对象内部使用Map类来保存数据，因此保存数据的格式为 “Key/value”。 session对象的value可以使复杂的对象类型，而不仅仅局限于字符串类型。

4、application对象

application 对象可将信息保存在服务器中，直到服务器关闭，否则application对象中保存的信息会在整个应用中都有效。与session对象相比，application对象生命周期更长，类似于系统的“全局变量”。

5、out 对象

out 对象用于在Web浏览器内输出信息，并且管理应用服务器上的输出缓冲区。在使用 out 对象输出数据时，可以对数据缓冲区进行操作，及时清除缓冲区中的残余数据，为其他的输出让出缓冲空间。待数据输出完毕后，要及时关闭输出流。

6、pageContext 对象

pageContext 对象的作用是取得任何范围的参数，通过它可以获取 JSP页面的out、request、reponse、session、application 等对象。pageContext对象的创建和初始化都是由容器来完成的，在JSP页面中可以直接使用 pageContext对象。

7、config 对象

config 对象的主要作用是取得服务器的配置信息。通过 pageConext对象的 getServletConfig() 方法可以获取一个config对象。当一个Servlet 初始化时，容器把某些信息通过 config对象传递给这个 Servlet。 开发者可以在web.xml 文件中为应用程序环境中的Servlet程序和JSP页面提供初始化参数。

8、page 对象

page 对象代表JSP本身，只有在JSP页面内才是合法的。 page隐含对象本质上包含当前 Servlet接口引用的变量，类似于Java编程中的 this 指针。

9、exception 对象

exception 对象的作用是显示异常信息，只有在包含 isErrorPage=”true” 的页面中才可以被使用，在一般的JSP页面中使用该对象将无法编译JSP文件。excepation对象和Java的所有对象一样，都具有系统提供的继承结构。exception 对象几乎定义了所有异常情况。在Java程序中，可以使用try/catch关键字来处理异常情况； 如果在JSP页面中出现没有捕获到的异常，就会生成 exception 对象，并把 exception 对象传送到在page指令中设定的错误页面中，然后在错误页面中处理相应的 exception 对象。

15.jsp的四大作用域

1.pageContext：页面域

页面作用域仅限于当前页面对象，可以近似于理解为java的this对象，离开当前JSP页面（无论是redirect还是forward），则pageContext中的所有属性值就会丢失。

2.request：请求域

请求作用域是同一个请求之内，在页面跳转时，如果通过forward方式跳转，则forward目标页面仍然可以拿到request中的属性值。如果通过redirect方式进行页面跳转，由于redirect相当于重新发出的请求，此种场景下，request中的属性值会丢失。

3.session：会话域

会话作用域是在一个会话的生命周期内，会话失效，则session中的数据也随之丢失。

4.application： 应用域

应用作用域是最大的，只要服务器不停止，则application对象就一直存在，并且为所有会话所共享。

16.session与cookie的区别

session与cookie都是会话跟踪技术，cookie通过在客户端记录信息确定用户身份。但是session的实现依赖于cookie。sessionId（session的唯一标识需要存放在客户端）

cookie与session的区别：

cookie的数据存放在客户端的浏览器上，而session数据存放在服务器上。

cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑到安全应用session。

session会在一定时间内保存在服务器上，当访问增多时，会比较占用服务器的性能，考虑到减轻服务器的性能方面的压力应用cookie。

单个cookie保存的数据不能超过4k，很多浏览器都限制一个站点最多保存20个cookie

所以我们通常把登录信息保存在session，其他需要保存的信息放在cookie

17.前端部分（了解即可）

html：超文本标记语言，定义页面的结构。

css：层叠样式表，用来美化页面。

javascript：主要用来验证表单做动态交互。

18.Ajax

Question1：什么是Ajax？

\*\*Answer1：\*\*Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。异步的javascript和xml。

Question2：作用？

Answer2： Ajax通过与服务器进行数据交换，Ajax可以使网页实现布局更新，这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

Question：怎么实现Ajax？

Answer3：使用XMLHttpRequest 可以异步向服务器发送请求，获取响应，完成布局更新。

使用场景：登录失败时不跳转页面，注册时提示用户名是否存在，二级联动等场景使用。

19.js和jQuery的关系

jQuery是一个js框架，封装了js的属性和方法，并增强了js的功能，让用户使用起来更加遍历。

原来使用js要处理很多兼容性的问题，由jQuery封装了底层，就不要处理兼容问题。并且原生的js的dom和事件绑定和Ajax等操作非常麻烦，jQuery封装后操作非常方便。

20.jQuery的常用选择器

元素选择器：通过元素名选取元素

#id 选择器：通过id获取一个元素

.class 选择器：通过类（css）获取元素

21.jQuery的Ajax和原生js实现的Ajax有什么关系？

jQuery中的Ajax也是通过原生的js封装的，封装完成后，让我们使用起来更加便利，不用去考虑底层实现和兼容性问题。

22.Html5

html5是最新版的html，是在原来的html4的基础上增加了一些标签例如画板，视频，web存储等高级功能。

23.CSS3

css3在原先的css2的基础上增加了一些功能例如：盒子功能、盒子和文字阴影、渐变、转换、移动、缩放等动画效果。

24.bootstrap

bootsrtap是一个移动设备优先的UI框架，我们可以不用写任何css，js代码就能实现比较漂亮的有交互性的页面。

25.框架部分

IT语境中的框架，特指为解决一个开放性问题而设计的具有一定约束性的支撑结构。在此结构上可以根据具体问题扩展、安插更多的组成部分，从而更迅速和方便地构建完整的解决问题的方案。

26.MVC模式

MVC是模型（model）– 视图（view）– 控制器（controller）的缩写。

最简单经典的是jsp（view）+ servlet（controller）+ JavaBean（model）

1.当控制器收到来气用户的请求

2.控制器调用JavaBean完成业务

3.完成业务后通过跳转jsp页面的方式给用户反馈信息

4.jsp给用户作出响应

控制器是核心

27.MVC框架

Question1：什么是MVC框架？

Answer1：是为解决传统MVC模式（jsp+servlet+JavaBean）所存在的问题而出现的框架。

传统MVC模式的问题：

所有的servlet和servlet映射都要配置在web.xml中，如果项目太大，web.xml就太庞大，不利于维护。

servlet的主要功能就是接受参数，调用逻辑，跳转页面，比如其他像字符编码，文件上传等功能也要写在servlet中，不能让servlet功能单一。

接受参数比较麻烦，不能通过model接收。

跳转页面方式比较单一（forward，redirect）。

28.Spring MVC执行流程

用户向服务器发送请求，请求被spring前端控制器DispatcherServlet捕获。（捕获）

DispaerServlet对请求url进行解析，得到请求资源标识符（URI），然后根据该URI调用HandlerMapping获得该Handler配置的所有相关对象（包括Handler对象以及Handler对象对应的拦截器），最后以HandlerExecutionChain对象的形式返回。（查找Handler）

DispatcherServlet根据获得的Handler，选择一个合适的HandlerAdapter，提取Request中的模型数据，填充Handler入参，开始执行Handler（Controller），Handler执行完成后，向DispatcherServlet返回一个ModelAndView对象。（执行Handler）

DispatcherServlet根据返回的ModelAndView，选择一个合适的ViewResolver（必须是已经注册到Spring容器中的ViewResolver）

通过ViewResolver结合Model和View，来渲染视图，DispaerServlet将渲染结果返回给客户端。（渲染视图）

核心控制器捕获请求，查找handler执行handler，选择ViewResolver，通过ViewResolver渲染视图并返回。

29.Spring MVC和Struts2的异同

核心控制器：Spring MVC的核心控制器是Servlet，而Struts2是Filter。

控制器实例：Spring MVC会比Struts2快一些，Spring MVC是基于方法设计，而Struts2是基于对象设计，每发一次请求都会实例一个action。

管理方式：大部分的公司的核心框架中，就会使用到Spring，而Spring MVC又是Spring中的一个模块，所以Spring对于Spring MVC的控制管理更加简单方便。

参数传递：Struts2中自身提供多种参数接受，其实都是通过ValueStack进行传递和赋值，而Spring MVC是通过方法的参数进行接收。

学习难度：Spring MVC 相较于Struts2更易学习。

interceptor的实现机制：Struts2有自己的interceptor机制，Spring MVC用的是独立的AOP方式。

Spring MVC处理Ajax请求直接通过返回数据，自动帮我们对象转换为JSON对象。而Struts2通过插件的方式进行处理。

30.Spring

Spring是J2EE应用程序框架，是轻量级的IOC和AOP的容器框架，主要是针对JavaBean的生命周期进行管理的轻量级框架。

IOC或DI

IOC控制反转。原先等service层需要调用Dao层时，Service就需要创建Dao层的一个实例。而运用Spring框架时，当Service层需要Dao层的一个实例时，Spring容器就会自行注入，不需要自行创建，降低耦合。

核心原理：配置文件+反射+容器（map）

1

AOP

面向切面编程

核心原理：使用动态代理的设计模式执行方法，执行方法前后或出现异常后加入相应的逻辑。

1

应用：

1.事务处理。执行方法前开启事务，执行完成后关闭事务，出现异常后回滚事务

2.权限判断。在执行方法前，判断是否具有权限。

3.日志。在执行前进行日志处理。

31.Spring事务的传播特性

事务传播行为类型 说明

PROPAGATION\_REQUIRED 如果当前没有事务，就新建一个事务，如果已经存在一个事务中，加入到这个事务中。这是最常见的选择。

PROPAGATION\_SUPPORTS 支持当前事务，如果当前没有事务，就以非事务方式执行。

PROPAGATION\_MANDATORY 使用当前的事务，如果当前没有事务，就抛出异常。

PROPAGATION\_REQUIRES\_NEW 新建事务，如果当前存在事务，把当前事务挂起。

PROPAGATION\_NOT\_SUPPORTED 以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

PROPAGATION\_NEVER 以非事务方式执行，如果当前存在事务，则抛出异常。

PROPAGATION\_NESTED 如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则执行与 PROPAGATION\_REQUIRED 类似的操作。

32.ORM

对象关系映射（Object Relational Mapping）模式是一种为了解决面向关系数据库存在的互不匹配的现象的技术。

ORM的方法基于三个核心原则：

简单：以最基本的形式建模数据。

传达性：数据库结构被任何人都能理解的语言文档化。

精确性：基于数据模型创建正确标准化了的结构。

33.MyBatis和Hibernate有什么异同

相同点：都是Java中orm框架，屏蔽JDBC API的底层访问细节，使我们不用与JDBC API打交道就可以完成数据库的持久化操作。

Mybatis的好处：屏蔽了JDBC API的底层访问细节，将sql语句与Java代码进行分离，提供了将结果集自动封装称为实体对象和对象的集合的功能。queryForList返回对象集合，用queryForObject返回单个对象，提供了自动将实体对象的的属性传递给sql语句的参数。

Hibernate的好处：Hibernate是一个全自动的orm映射工具，它可以自动生成sql语句，并执行返回Java结果。

1

2

3

4

不同点：

1.hibernate要比mybatis功能强大很多，因为hibernate会自动生成sql语句。

2.mybatis需要我们自己在xml配置文件中写sql语句，hibernate无法直接控制该语句，我们就无法写特定的高效率的sql语句。

3.mybatis要比hibarnate简单的多，mybatis是面向sql的，不用考虑对象间一些复杂的映射关系。

34.webService

webservice是一个SOA（面向服务编程），它是不依赖于语言，不依赖于平台，可以实现不同的语言间的相互调用。通过Internet进行基于Http协议的网络应用的交互。

1. 异构系统的整合（不同语言）

2. 不同客户端的整合，浏览器、手机端（Android、iOS）、微信端、PC端等终端的访问。

3. 例子：天气预报，通过实现webservice的客户端调用远程服务实现的。

单点登录：一个服务实现所有系统的登录。

35.activiti

activiti是一个业务流程管理（BPM）和工作刘系统，适用于开发人员和管理员，其核心是超快速、稳定的BPMN2流程引擎，它易于与spring集成使用。

主要用于在OA中，把线下流程放到线上来，把现实生活中一些流程图定义到系统中，然后通过输入表单数据完成业务。

它主要用在OA系统的流程管理中。

例：请假流程：小于三天–>一级主管审批，大于三天–>二级主管审批。

36.Linux

Linux事业个长时间运行比较稳定的操作系统，所以我们一般用它作为服务器，Linux本身具有c的编译环境，我们的一些软件是没有安装包的（例如redis），需要在Linux编译得到软件包。

常用的命令

pwd 获取当前路径

cd 跳转到指定目录

su 切换用户身份

ls 列举当前目录的文件和文件夹

文件操作命令

cat 查看

rm -rf 删除

vi 修改

文件夹的操作

mkdir 创建文件夹

rm-rf 删除

1

2

3

4

5

6

7

8

9

10

11

12

13

Question：如何远程连接Linux？

Answer：需要使用Linux服务器anzhuangssh服务端，且ssh服务端的端口号为22，之后使用ssh客户端连接服务器。常用的ssh客户端又Xshell、Putty。

Question2：云主机是什么？

Answer2：云主机就是一些云服务运营商（例如阿里、华为）提供的远程的服务器的功能，我们开发者只需按照自己的需求购买即可。

37.数据库分类和常用的数据库

数据库的分类：

1.关系型数据库（mysql、oracle、SQL server）

2.非关系型数据库（redis、mogodb、Hadoop）

1

2

关系数据库的三范式：

范式就是就是规范，关系型数据库在设计表时要遵循的三个规范。

1

第一范式（无重复的列） 定义：数据库表的每一列都是不可分割的原子数据项，而不能是集合，数组，记录等非原子数据项。如果实体中的某个属性有多个值时，必须拆分为不同的属性 通俗解释：一个字段只存储一项信息 eg:班级：高三年1班，应改为2个字段，一个年级、一个班级，才满足第一范式，否则不满足第一范式。

学号 姓名 班级

0001 小红 高三年1班

改成

学号 姓名 年级 班级

0001 小红 高三年 1班

1

2

3

4

5

第二范式（属性完全依赖于主键） 定义：满足第一范式前提，当一个主键由多个属性共同组成时，才会发生不符合第二范式的情况。比如有两个属性的主键，不能存在这样的属性，它只依赖于主键中的一个属性，这就是不符合第二范式 通俗解释：任意一个字段都只依赖表中的同一个字段

eg:比如不符合第二范式

学生证名称 学生证号 学生证办理时间 借书证名称 借书证号 借书证办理时间

改成2张表如下

学生证表

学生证名称 学生证号 学生证办理时间

借书证表

借书证 借书证号 借书证把你拉时间

1

2

3

4

5

6

7

第三范式（属性不能传递依赖于主属性） 定义：满足第二范式前提，如果某一属性依赖于其他非主键属性，而其他非主键属性又依赖于主键，那么这个属性就是间接依赖于主键，这被称作传递依赖于主属性。 通俗理解：一张表最多只存2层同类型信息

eg:爸爸资料表，不满足第三范式

爸爸 儿子 女儿 女儿的小熊 女儿的海绵宝宝

改成 爸爸信息表：

爸爸 儿子 女儿

女儿信息表

女儿 女儿的小熊 女儿的海绵宝宝

1

2

3

4

5

6

反三范式：有时候为了效率，可以设置重复的字段

38.事务的四大特征

事务是并发控制的单位，是用户定义的一个操作序列，这些操作要么都做，要么都不做，是一个不可分割的工作单位。

事务的四大特征：

原子性：表示事务内操作不可再分割，要么都成功，要么都失败。

一致性：要么都成功，要么都失败，后面的失败了，要对前面的操作进行回滚。

隔离性：一个事务开始后，不能受其他事务干扰。

持久性：表示事务开始了，就不能终止。

39.mysql数据库的默认的最大连接数

Question：为什么需要最大连接数？

Answer：特定服务器上的数据库只能支持一定数目同时连接，这时候我们设置最大连接数（最多同时服务多少连接）。在数据库安装时都会有一个最大的连接数。mysql默认的最大连接数是100；

40.mysql和oracle的分页语句

Question1：为什么需要分页？

Answer1：在查询很多数据时，不可能完全显示数据所以要进行分页显示。

Mysql使用关键字limit来进行分页的，limit offset，size表示从多少索引去多少位

1 SELECT

2 \*

3 FROM

4 student

5 LIMIT (PageNo - 1) \* PageSize,PageSize;

1

2

3

4

5

oracle的分页有点记不住了，记得是嵌套查询………我去查了一下

1 SELECT

2 \*

3 FROM

4 (

5 SELECT

6 ROWNUM rn ,\*

7 FROM

8 student

9 WHERE

10 Rownum <= pageNo \* pageSize

11 )

12 WHERE

13 rn > (pageNo - 1) \* pageSize

1

2

3

4

5

6

7

8

9

10

11

12

13

41.触发器的使用场景

触发器需要触发条件，当条件满足以后做什么操作。

create trigger triggerName

after/before insert/update/delete on 表名

for each row #这句话在mysql是固定的

begin

sql语句;

end;

1

2

3

4

5

6

7

8

9

10

11

下单操作

DELIMITER $$

CREATE TRIGGER `test`.`trigger\_insert` AFTER INSERT ON orders

FOR EACH ROW

BEGIN

UPDATE goods SET num=num-new.much WHERE id=new.goods\_id;

END$$

DELIMITER ;

1

2

3

4

5

6

7

42.存储过程

优点：

1.允许模块化程序设计，就是说只需要创建一次过程，以后在程序中就可以调用该过程任意次。

2.允许更快执行，如果某操作需要执行大量SQL语句或重复执行，存储过程比SQL语句执行的要快。

3.减少网络流量，例如一个需要数百行的SQL代码的操作有一条执行语句完成，不需要在网络中发送数百行代码。

4.更好的安全机制，对于没有权限执行存储过程的用户，也可授权他们执行存储过程。

43.JDBC

JDBC（Java Data Base Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。

Java通过定义接口，让数据库厂商自己实现接口，对于我们开发者而言，只需导入对应厂商开发的实现即可。

以下为之前的一个JDBC实例部分：

public static Connection getConnection() {

Connection conn = null;

try {

// 1.注册驱动

Class.forName("com.mysql.jdbc.Driver");

// 2.建立连接

conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/Fish?useSSL=yes", "root", "1234");

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

System.out.println("注册驱动失败");

e.printStackTrace();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return conn;

}

/\*\*

\* save函数，将源Date与状态存入数据库

\*

\* @param strDate

\* @param state

\*/

public static void save(String strDate, String state) {

PreparedStatement ps = null;

Connection conn = null;

String sql = "insert into fish(date,state) values(?,?)";

try {

conn = getConnection();

ps = conn.prepareStatement(sql);

ps.setString(1, strDate);

ps.setString(2, state);

ps.executeUpdate();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally {

try {

if (conn != null) {

conn.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

44.JDBC中PrepareStatement相比Statement的好处

大多数情况下我们都使用PrepareStatement

PrepareStatement是预编译的，比Statement速度更快。

代码的可读性增强。

安全性，PrepareStatement可以防止SQL注入攻击。

45.数据库连接池的作用

1、资源复用

由于数据库连接得到重用，避免了频繁创建、释放连接引起的大量性能开销。在减少系统消耗的基础上，另一方面也增进了系统运行环境的平稳性（减少内存碎片以及数据库临时进程/线程的数量）。

2、更快的系统响应速度

数据库连接池在初始化过程中，往往已经创建了若干数据库连接至于池中备用。此时连接的初始化工作均已完成。对于业务请求处理而言，直接利用现有可用连接，避免了数据库连接初始化和释放过程的时间，从而缩减了系统整体响应时间。

3、统一的连接管理，避免数据库连接泄漏在较为完备的数据库连接池实现中，可根据预先的连接占用超时设定，强制收回被占用连接。从而避免了常规数据库连接操作中可能出现的资源泄漏。

46.数据库优化方面的事情

查找、定位慢查询、并优化

创建索引:创建合适的索引，我们就可以现在索引中查询，查询到以后直接找对应的记录。

分表 ：当一张表的数据比较多或者一张表的某些字段的值比较多并且很少使用时，采用水平分表和垂直分表来优化

读写分离：当一台服务器不能满足需求时，采用读写分离的方式进行集群。

缓存:使用 redis来进行缓存等等

47.查找满查询并定位慢查询

在项目自验后转测试之前，在启动mysql数据库时开启慢查询，并且把执行慢的语句写到日志中，在运行一段时间后，通过查看日志，找到慢查询的语句。

使用explain慢查询的语句来详细分析语句的问题。

48.选择合适的数据库引擎

在开发中，我们经常使用MyISAM/INNODB/Memory

MyISAM：如果对事务要求不高，同时是以查询和添加为主的，我们考虑使用MyISAM存储引擎，比如bbs中的发帖表和回复表

INNODB：对事务要求高，保存的数据都是重要的数据，我们建议使用INNODB，比如订单表，账号表。

Memory：数据变化频繁，不需要入库，同时又频繁的查询和修改，我们考虑使用memory，速度极快。

MyISAM和INNODB的区别：

事务安全：MyISAM不支持事务，INNODB支持。

查询和添加速度：MyISAM不用支持事务，就不用考虑同步锁，

支持全文索引：MyISAM支持，INNODB不支持

锁机制：MyISAM支持表锁，INNODB支持行锁（事务）

外键：MyISAM不支持外键，INNODB支持外键（通常不设置外键，通常在程序中保证数据的一致）

49.选择合适的索引

索引（index）是帮助DBMS高效获取数据的数据结构。

分类：普通索引、唯一索引、主键索引、全文索引

普通索引：允许重复的值出现

唯一索引：除了不能有重复的记录外，其他和普通索引一样

主键索引：是随着设定主键而创建的也就是把某个列设为主键的时候，数据库就会给该列创建索引，这就是主键索引，唯一且没有null值。

全文索引：用来对表中的文本域（char、varchar、text）进行索引。

50.索引使用小技巧

索引的弊端：

占用磁盘空间

对dml（插入、修改、删除）操作有影响，变慢。

使用场景：

经常使用where条件查询

该字段的值不是唯一的几个值（sex）

字段内容不是频繁变化

具体技巧：

对于创建的多列索引（复合索引），不是使用的第一部分就不会使用索引

alter table dept add index my\_ind (dname,loc); //dname左边的列，loc就是右边的列

explain select \* from dept where dname="aaa"; //会使用索引

explain select \* from dept where loc="aaa"; //不会使用索引

1

2

3

对于使用like的查询，查询如果是 ‘%aaa’ 不会使用索引，’aaa%’ 会使用索引

explain select \* from dept where dname like '%aaa'; //不使用索引

explain select \* from dept where dname like 'aaa%'; //使用索引

1

2

如果条件中有or，有条件没有使用索引，即使其中有条件带索引页不会使用，换言之，就是要求使用的所有字段，都必须单独使用时能使用索引。

如果列类型是字符串，那一定要早条件中将数据使用引号引起来，否则不使用索引。

如果mysql使用全表扫描要比使用索引快，则不使用索引。例如，表里只有一条数据。

51.数据库优化之分表

分表分为水平分表（按行）和垂直分表（按列）：

根据经验，mysql表数据一般达到百万级别，查询效率会很低，容易造成表锁，甚至堆积很多连接，直接挂掉。水平分表能够很大程度减少这些压力。按行数据进行分表。

如果一张表中的某个字段非常多（长文本、二进制等）而且只有在很少的情况下才会查询。（例如考试详情，我们一般只关注分数，不关心详情），这时候就可以把字段多的单独放到一个表，通过外键关联起来。

水平分表：

按时间分表

按区间范围进行分表

例如：table\_1 user\_id 从 1～100w

table\_2 user\_id 从 100w～200w

table\_3 user\_id 从 200w～300w

1

2

3

hash分表

通过一个原始目标的id或者通过一定的hash所发计算出存储表的表名，然后访问响应的表。

1

52.数据库优化之读写分离

一台数据库支持的最大并发连接数是有限的，如果用户并发访问太多，一台服务器满足不了要求时，就可以集群处理，mysql的集群处理技术最常用的就是读写分离。

主从同步

数据库最终会把数据库持久化到磁盘，如果集群必须确保每个数据库服务器的数据是一致的，能改变数据库数据的操作都往主数据库上写，而其他的数据库从主数据库上同步数据。

1

读写分离

使用负载均衡来实现写的操作都往主数据库去， 而读的操作都往从服务器去。

1

53.数据库优化之缓存

在持久层（dao）和数据库（db）之间添加一个缓存层，如果用户访问的数据已经缓存起来时，在用户访问时直接从缓存中获取不用访问数据库。而缓存是在操作内存级，访问速度很快。

作用：减少数据库服务器驱动，减少访问时间

Java中常用的缓存有：

1.hibernate的二级缓存，该缓存不能完成分布式缓存。

2.可以使用redis来作为中央缓存，对缓存的数据进行集中处理。

1

2

54.sql语句优化小技巧

DDL（数据定义语言）优化：

1.通过禁用索引来提供导入数据性能，这个操作主要针对有数据库的表追加数据。

2.关闭唯一校验：set unique\_check = 0 //关闭

3.修改事务提交方式（导入）变多次提交为一次提交：set autocommit = 0 //关闭

1

2

3

4

5

DML（数据操纵语言）优化：变多次提交为一次

insert into test values(1,2);

insert into test values(1,3);

insert into test values(1,4);

........

//合并多条为一条

insert into test values(1,2) (1,3) (1,4);

1

2

3

4

5

6

DQL（数据查询语言）优化：

order的优化：

1.多用索引排序

2.普通结果排序

1

2

3

55.JDBC批量插入几百万条数据怎么实现

变多次提交为一次

使用批量操作

能省的时间非常可观，像这样的批量插入操作能不使用代码就不使用，可以使用存储过程来实现

56.redis

Question1： redis是什么？

Answer1： redis是一个key-value的mysql数据库，先到内存中根据一定的策略持久化到磁盘，即使断电也不会丢失数据，支持的数据类型比较多。主要用来做缓存数据库的数据和web集群当作中央缓存存放session

Question2： 简单说一下redis的使用场景？

Answer2： 首先redis可以用做缓存，把经常需要查询的，很少修改的数据放到读速度很快的空间（内存），以便下次访问减少时间，减轻压力，减少访问时间。其次是计数器，redis中的计数器是原子性的内存操作，可以解决库存溢出问题，进销存系统内存溢出。redis还可作为session缓存服务器，web集群时作为session缓存服务器。

57.redis存储对象的方式

JSON字符串：需要把对象转换为json字符串，当做字符串处理，直接使用set、get来设置或者获取。优点：设置和获取比较简单。缺点：没有提供专门的方法，需要把对象转换为json（jsonlib）

字节：需要做序列化，就是把对象序列化为字节保存

如果担心json转对象比较消耗资源的情况，这个问题需要考虑几个方面：

就是使用的json转换lib是否存在性能问题

即使数据的数量级别，如果是存储百万级别的大数据对象，建议采用存储序列化对象方式，如果少量的数据级对象。或者是数据对象字段不多，还是建议采用json准还成String的方式

58.redis数据淘汰机制

Question： 为什么要淘汰？

Answer： 内存有限，需要保存有效的数据。

volatile-lru：使用LRU算法进行数据淘汰（淘汰上次使用时间最早的，且使用次数最少的key），只淘汰设定了有效期的key

allkeys-lru：使用LRU算法进行数据淘汰，所有的key都可以被淘汰

volatile-random：随机淘汰数据，只淘汰设定了有效期的key

allkeys-random：随机淘汰数据，所有的key都可以被淘汰

volatile-ttl：淘汰剩余有效期最短的key

59.java访问redis及redis集群

使用jedis java客户端来访问redis服务器，有点类似于jdbc访问mysql一样。

如果是Spring进行集群时，可以使用Spring data 来访问redis，spring data是对jedis的二次封装，类似于jdbcTemplate 和JDBC之间的关系

redis集群：当一台数据无法满足需求时，可以使用redis集群来处理，类似于mysql的读写分离

————————————————

版权声明：本文为CSDN博主「小过121」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_40422256/article/details/81813096