前言

这本书其实是我的一本笔记（还在整理中）. 我是也是刚找到工作. 这本笔记主要记录了我之前面试遇到的问题以及我在网上整理的一些资料 主要是面向 junior 级别的 就是我们这些小菜鸟啦 ~

目前只有我一个人 希望大家都能加入进来一起贡献或者是指出错误

如果有确实帮助到大家 可以进我的源代码在右上角 star 一下下 :)

Gitbook地址

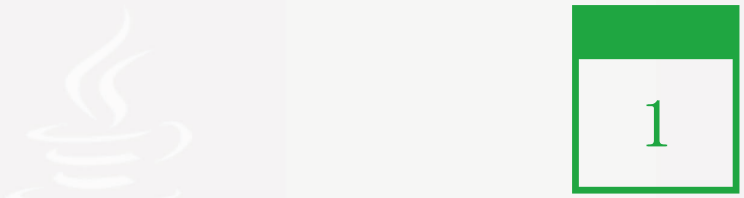
Github托管地址

如何贡献

任何问题都欢迎直接联系我 dongchuan55@gmail.com QQ 526402328

因为现在 gitbook 支持在线编辑功能, 类似于 office word. 所以想要贡献的同学可以直接联系我开权限!

目录



1

Java

第一部分是 Java 的基础面试题 补充ing

说一说 Java

Sun 公司在 1995 创建

Java 的一些特点?

- Object Oriented 面向目标的
- Platform Independent 平台独立
- Interpreted 解释性语言
- Multi-threaded 多线程

但是 Java 最重要的特点是**平台独立**

平台独立意味着我们可以在一个系统编译它然后在另外一个系统使用它

PS : 有一些中文的博客会说 java 也是编译性语言 因为国内博客都是你抄我 我抄你 所以你懂的 当然最好就是给出编译混合解释性这个说法 然后给出自己的理解 我自己在国外面试的时候也问了几个面试官关于这个问题 都说解释性语言是最准确的.

JDK JRE JVM?

- 解释它们的区别
- 为什么 JVM 不是平台独立的

JDK

Java Development Kit 用作开发, 包含了JRE, 编译器和其他的工具(比如: JavaDoc，Java调试器), 可以让开发者开发、编译、执行Java应用程序.

JRE

Java 运行时环境是将要执行 Java 程序的 Java 虚拟机, 可以想象成它是一个容器, JVM 是它的内容.

JRE = JVM + Java Packages Classes(like util, math, lang, awt,swing etc)+runtime libraries.

JVM

Java virtual machine (Java 虚拟机) 是一个可以执行 Java 编译产生的 Java class 文件 (bytecode) 的虚拟机进程, 是一个纯的运行环境.

JVM 不是平台独立的

Java被设计成允许应用程序可以运行在任意的平台, 而不需要程序员为每一个平台单独重写或者是重新编译. Java虚拟机让这个变为可能, 因为它知道底层硬件平台的指令长度和其他特性.

Reference

Diff between JRE and JVM

什么是对象 (Object)?

- 对象是程序运行时的实体
- 它的状态存储在 fields (也就是变量)
- 行为是通过方法 (method) 实现的
- 方法上操作对象的内部的状态
- 方法是对象对对象的通信的主要手段

封装 Encapsulation

- 使一个类的变量 private
- 提供 public 方法来调用这些变量. 所以外部类是进不去的. 这些变量被隐藏在类里了. 只能通过已经定义的 pulic 方法调用.

好处

当我们修改我们的实现的代码时, 不会破坏其他调用我们这部分代码的代码. 可维护性, 灵活性和可扩展

多态 Polymorphism

多态就是指一个变量, 一个方法或者一个对象可以有不同的形式.

多态主要分为

- 重载overloading

  就是一个类里有两个或更多的函数，名字相同而他们的参数不同.

- 覆写overriding

  是发生在子类中！也就是说必须有继承的情况下才有覆盖发生. 当你继承父类的方法时, 如果你感到哪个方法不爽，功能要变，那就把那个函数在子类中重新实现一遍.

例子

重载 静态捆绑 (static binding) 是 Compile 时的多态

```
EmployeeFactory.create(String firstName, String lastName){...}
EmployeeFactory.create(Integer id, String lastName){...}
```

覆写 动态捆绑 (dynamic binding) 是 runtime 多态

```
public class Animal {
    public void makeNoise()
    {
        System.out.println("Some sound");
    }
}

class Dog extends Animal{
    public void makeNoise()
    {
        System.out.println("Bark");
    }
}

class Cat extends Animal{
    public void makeNoise()
    {
        System.out.println("Meawoo");
    }
```

```
}


public class Demo
{
    public static void main(String[] args) {
        Animal a1 = new Cat();
        a1.makeNoise(); //Prints Meowoo

        Animal a2 = new Dog();
        a2.makeNoise(); //Prints Bark
    }
}
```

接口 Interface

接口是抽象方法的集合。一个类实现一个或多个接口，因此继承了接口的抽象方法.

接口的特点

- 不能实例化
- 没有构造体
- 所有方法都是抽象的 (abstract).同时也是隐式的 public static. 也就是说声明时, 可以省略 public static.
- 只能含有声明为 final static 的 field

Static 关键字

Static 关键字表明一个成员变量或者是成员方法可以在没有所属的类的实例的情况下直接被访问

声明为 **static 的方法**有以下几条限制：

1. 仅能调用其他的 static 方法
2. 只能访问 static 变量.
3. 不能以任何方式引用 this 或 super
4. 不能被覆盖.

声明为 **static 的变量**实质上就是全局变量. (+ final 就是全局**常**量). 当声明一个对象时, 并不产生 static 变量的拷贝, 而是该类所有的实例变量共用同一个 static 变量.

对于静态类，只能用于嵌套类内部类中。

Reference

- Static class in Java - GeeksforGeeks

Singleton 单例模式

Java中单例模式定义：“一个类有且仅有一个实例，并且自行实例化向整个系统提供。”

```
public class Singleton {
    private Singleton() {
        // do something
    }
    private static class SingletonHolder {
        private static final Singleton INSTANCE = new Singleton();
    }
    public static final Singleton getInstance() {
        return SingletonHolder.INSTANCE;
    }
}
```

多选题注意

- 一是单例模式的类只提供私有的构造函数
- 二是类定义中含有一个该类的静态私有对象
- 三是该类提供了一个静态的公有的函数用于创建或获取它本身的静态私有对象。

Reference

- 深入浅出单实例Singleton设计模式 | 酷壳 - CoolShell.cn

equals() 与 hashcode()

**Equal **

如果需要比较对象的值，就需要equal方法了. 看一下JDK中equal方法的实现：

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

也就是说，默认情况下比较的还是对象的地址. 所以如果把对象放入Set中等操作，就需要重写eqaul方法了

重写之后的 equals() 比较的就是对象的内容了

**hashcode **

When inserting an object into a hastable you use a key. The hash code of this key is calculated, and used to determine where to store the object internally. When you need to lookup an object in a hashtable you also use a key. The hash code of this key is calculated and used to determine where to search for the object.

The hash code only points to a certain "area" (or list, bucket etc) internally. Since different key objects could potentially have the same hash code, the hash code itself is no guarantee that the right key is found. The hashtable then iterates this area (all keys with the same hash code) and uses the key's equals() method to find the right key. Once the right key is found, the object stored for that key is returned.

So, as you can see, a combination of the hashCode() and equals() methods are used when storing and when looking up objects in a hashtable.

If equal, then same hash codes too. Same hash codes no guarantee of being equal.

所有类的基类是哪个类?

java.lang.Object

Does Java support multiple inheritance?

Java doesn't support multiple inheritance.

Path 与 Classpath?

Path 和 Classpath 是操作系统的环境变量.

- Path 定义了系统可以在哪里找到可执行文件(.exe)
- classpath 定义了 .class 文件的位置.

反射机制

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；对于任意一个对象，都能够调用它的任意一个方法和属性；这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制.

主要作用有三：

1. 运行时取得类的方法和字段的相关信息。
2. 创建某个类的新实例(.newInstance())
3. 取得字段引用直接获取和设置对象字段，无论访问修饰符是什么。

用处如下：

1. 观察或操作应用程序的运行时行为。
2. 调试或测试程序，因为可以直接访问方法、构造函数和成员字段。
3. 通过名字调用不知道的方法并使用该信息来创建对象和调用方法。

final 关键字

final 类是不能被继承的 这个类就是最终的了 不需要再继承修改 比如很多 java 标准库就是 final 类

final 方法不能被子方法重写

final + static 变量表示常量

一个 .java 源文件是否可以包含多个类

可以得

但只能有一个是 public 的类 而且这个 public 类必须与文件名一样

& 与 &&

都可以表示逻辑与 and，但是 && 具有短路功能 第一个表达式错了 第二个就被忽略了。`&` 的表达式是先计算后求与。

除此外 & 可以用作位运算符

`|` 也有类似差异。

java - Difference between & and && - Stack Overflow

int 与 integer

int 是数据类型

interger 是 int 的封装类

int 默认值为 0 interger 默认值为 null 所以 interger 可以用来判断变量是否赋值 即 null 和 0 的区别

作用域的区别

作用域	当前类	同一个 package	子孙类	其他 package
public	0	0	0	0
protected	0	0	0	X
friendly	0	0	X	X
private	0	X	X	X

异常

异常是指java程序运行时（非编译）所发生的非正常情况或错误

Java使用面向对象的方式来处理异常，它把程序中发生的每个异常也都分别封装到一个对象来表示的，该对象中包含有异常的信息

Java对异常进行了分类，所有异常的根类为java.lang.Throwable

Throwable下面又派生了两个子类：Error和Exception

final, finally, finalize的区别

- final 用于声明属性,方法和类, 分别表示属性不可变, 方法不可覆盖, 类不可继承.
- finally 是异常处理语句结构的一部分，表示总是执行.
- finalize 是Object类的一个方法，在垃圾收集器执行的时候会调用被回收对象的此方法，可以覆盖此方法提供垃圾收集时的其他资源回收，例如关闭文件等. JVM不保证此方法总被调用.

Programme

Gabage Collection

什么是GC

GC是垃圾收集的意思(Gabage Collection), 内存处理是编程人员容易出现问题的地方, 忘记或者错误的内存回收会导致程序或系统的不稳定甚至崩溃, Java提供的GC功能可以自动监测对象是否超过作用域从而达到自动回收内存的目的, Java语言没有提供释放已分配内存的显示操作方法.

垃圾回收器的基本原理是什么?

当程序员创建对象时，GC就开始监控这个对象的 地址、大小以及使用情况. 通常，GC采用有向图的方式记录和管理堆（heap）中的所有对象。通过这种方式确定哪些对象是"可达的"， 哪些对象是"不可达的".当GC确定一些对象为"不可达"时(比如设置为 null)，GC就有责任回收这些内存空间.

有什么办法主动通知虚拟机进行垃圾回收?

可以.程序员可以手动执行System.gc(),通知GC运行,但是Java语言规范并不保证GC一定会执行. 这个选择题的时候有考.

字节流与字符流

- 字节流继承于InputStream OutputStream
- 字符流继承于InputStreamReader OutputStreamWriter

字符流使用了缓冲区 (buffer),而字节流没有使用缓冲区

底层设备永远只接受字节数据

字符是字节通过不同的编码的包装

字符向字节转换时，要注意编码的问题

Collection

Collection 的子类是 List 和 Set

Multi-Thread

Visitor Pattern

Visitor -- switch case

Problem on chain

序列化 serialization

Serialization is a mechanism by which you can save the state of an object by converting it to a byte stream.

JAVA中实现serialization主要靠两个类:

- ObjectOuputStream
- ObjectInputStream

他们是JAVA IO系统里的OutputStream和InputStream的子类

自定义序列化的作用如下:

1. Persist only meaningful data.
2. Manage serialization between different versions of your class.
3. Avoid exposing the serialization mechanism to client API.

Reference

- The Java HotSpot: Customizing Java Serialization [Part 2]

Initialization and Cleanup

5. 初始化和清理

5.7 构造器初始化

5.7.1 初始化顺序

类内部变量定义的先后顺序决定了其初始化的顺序，并且会在任何方法（包括构造器，与顺序无关）被调用之前也会得到初始化。对于静态对象与非静态对象：先初始化静态对象，然后是非静态对象。

5.7.2 静态数据的初始化

静态数据只占用一份存储区域，static 关键字不能用于局部变量，因为它只能作用于域。如果一个域是静态的基本类型域且未对其初始化，那么它就会获得基本类型的标准初值；如果是一个对象引用，则初始化为 null.

静态初始化只有在必要时才会进行，且只被初始化一次，即如果不创建相应的对象或是引用相应的静态对象，那么则不会被初始化。

对象创建过程：

1. 构造器实际上也是静态方法。Java 解释器首先查找类路径定位相应 class 文件。
2. 载入 class 文件，执行静态初始化，静态初始化只在类对象首次加载的适合进行一次。
3. 使用 new 创建对象时首先将在堆上为对象分配足够的存储空间。
4. 存储空间清零，故其所有基本类型数据置为默认值。
5. 执行所有定义处的初始化动作。
6. 执行构造器。

Java Data Types - Java 数据类型

JVM 可以操作的数据类型分为两类: primitive types 和 reference types. 类型检查通常在编译期完成，不同指令操作数的类型可以通过虚拟机的字节码指令本身确定。

Primitive type

JVM 所支持的基本数据类型有：数值类型(Numeric types)，布尔类型(Boolean type) 和 returnAddress 类型。其中数值类型又可以分为整型和浮点型两种。

- 整型：byte(8 bit)，short(16 bit)，int(32 bit)，long(64 bit)，char(16 bit unsigned)
- 浮点型：float(32 bit)，double(64 bit)
- 布尔型：boolean 通常用 int 型表示，Oracle 中用 byte 表示
- returnAddress：一条字节码指令的操作码

Reference type

引用类型分为三种：Class Types，Array Types 和 Interface Types，这些引用类型的值分别由类实例、数组实例和实现了某个接口的类实例或者数组实例动态创建。引用类型中有一特殊的值 `null`，引用类型的默认值就是 null.

形式参数传递

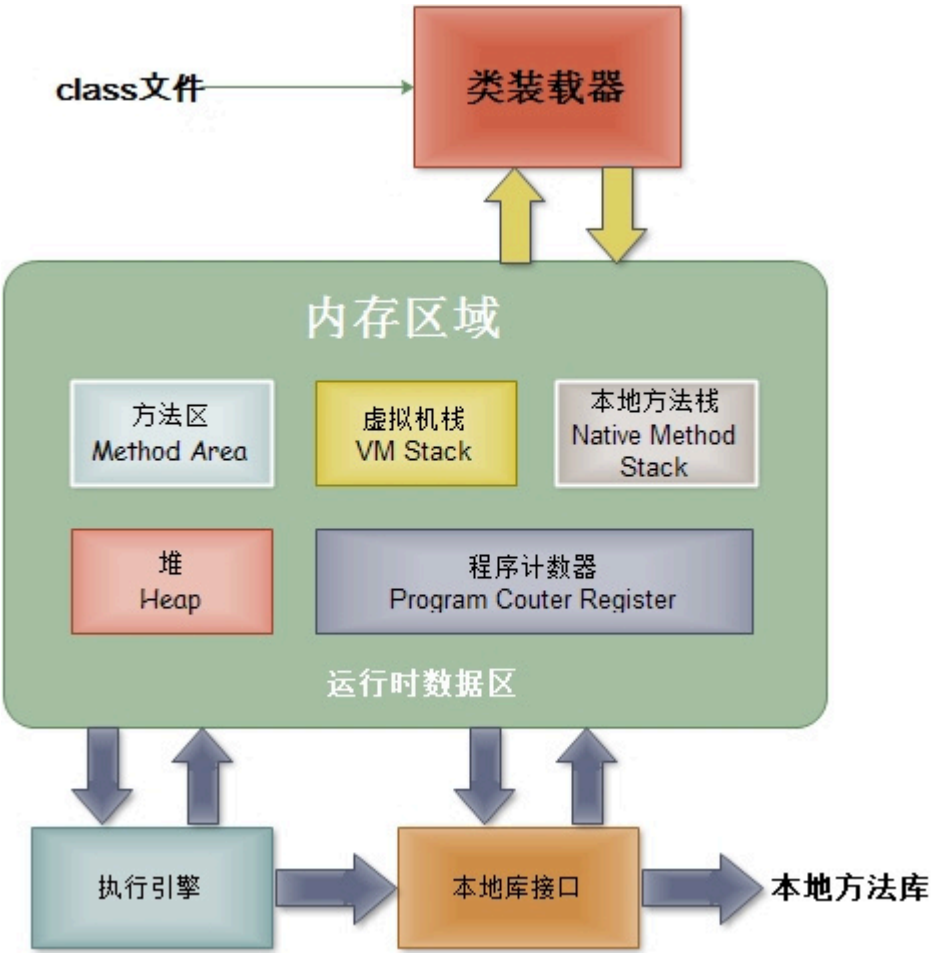
基本类型作为形式参数传递不会改变实际参数，引用类型作为形式参数传递会改变实际参数。JDK1.5之后含有基本类型的包装类型，即自动拆装箱的功能，故将基本类型的相应对象作为参数传递时会自动拆箱为基本类型，故也不改变实际参数的值。

Reference

- Chapter 2. The Structure of the Java Virtual Machine

Run-Time Data Areas - 运行时数据区域

JVM 运行时会有几个运行时数据区域，如下图所示。



图片 1.1 Run Time Data Areas

The pc Register - 程序计数器

线程私有内存，保存**当前线程所执行的字节码的行号指示器**，这里和计算机组成原理中的计数器不太一样，计组中的 PC 指的是**下一条要执行的指令**的地址。JVM 中常有多个线程执行，故每条线程都需要有一个独立的程序计数器。

如果线程执行的是 Java 方法，哪儿计数器记录的就是正在执行的虚拟机字节码指令的地址；如果执行的是 Native 方法，这个计数器则为空。

P.S. 这块内存无`OutOfMemoryError`

Java Virtual Machine Stacks - Java 虚拟机栈

线程私有，虚拟机栈描述的是 Java 方法执行的内存模型，每个方法在执行时会创建一个栈帧，栈帧中保存有局部变量表、操作数栈、动态链接和方法出口等。粗略来讲 Java 内存区分为堆和栈，实际上『栈』指的往往是虚拟机栈中的局部变量表部分。

局部变量表中存放了编译期可知的各种基本数据类型、对象引用类型和 returnAddress 类型。方法运行期间局部变量表大小不变。

Native Method Stacks - 本地方法栈

和虚拟机栈类似，不过区别在于虚拟机栈为 Java 方法（字节码）服务，而本地方法栈为 Native 方法服务（类似 C 语言中的栈）。具体实现可将这两者合二为一。

Heap - 堆

堆是被所有线程共享的一块内存区域。一般来说所有的对象实例和数组都要在堆上分配，但一些优化技术导致不一定所有对象实例都在堆上分配。

Method Area - 方法区

各线程共享的一块内存区域，和操作系统中进程中的『文本段』有些类似，用于存储虚拟机加载的类信息、常量、静态常量和即时编译器编译后的代码数据等。

Run-Time Constant Pool - 运行时常量池

这一部分是方法区的一部分，用于保存 Class 文件中编译期生成的字面值和符号引用。

直接内存

这一部分并不是虚拟机运行时的数据区域，用于 Native 函数分配堆外内存，提高性能用（不必在操作系统堆和 Java 堆复制数据）。

Reference

- 《深入理解 Java 虚拟机》
- Java 内存区域详解 - SegmentFault
- Chapter 2. The Structure of the Java Virtual Machine

2

What are considered as a web component?


HTML
unity
HTML

什么是 J2EE?

An environment for developing and deploying enterprise applications.

The J2EE platform consists of a set of services, application programming interfaces (APIs), and protocols that provide the functionality for developing multitier, web-based applications.

J2EE 应用的四个部分?

我更喜欢说是层

- 客户端层
- web层 (Servlet and JSP)
- 业务层 (JavaBeans)
- 企业信息系统层 (Enterprise Information System tier) 或者叫Resource adapter 包含数据库等

J2EE客户端有哪些类型？

- Applets
- Application clients
- Java Web Start-enabled clients, by Java Web Start technology.
- Wireless clients, based on MIDP technology.

Hibernate是什么?

- object-relational
-
- In hibernate we can write HQL instead of SQL which save developers to spend more time on writing the native SQL.
- 我们能像处理 Java 对象一样处理数据库
- 所以可以在处理的时候加入 Java 语言的特性 比如继承啊 多态啊 。。
- Hibernate also allows you to express queries using java-based criteria .

什么是事务 - transaction

事务是应用程序中一系列严密的操作，所有操作必须成功完成，否则在每个操作中所做的所有更改都会被撤消。例如，将资金从支票帐户转到储蓄帐户中是一项事务，按步骤如下进行：

检查支票帐户是否有足够的资金来支付此转帐操作。

如果支票帐户中有足够的资金，则将该笔资金记入此帐户的借方。

将这些资金记入储蓄帐户的贷方。

将此次转帐记录到支票帐户日志中。

将此次转帐记录到储蓄帐户日志中。

什么是 servlet?

Servlets 是服务器端的部件

是纯的 java 对象

设计用于多种协议 特别是HTTP

JSP

JavaServer Pages (JSP)

delivering **dynamic content** to web applications in a portable, secure and well-defined way.

The JSP Technology allows us to use HTML, Java, JavaScript and XML in a single file to create high quality and fully functionaly User Interface components for Web Applications.

Ear, Jar 和 War 文件的区别?

- Jar files are intended to hold generic libraries of Java classes, resources, etc.
- War files are intended to contain complete Web applications.
- Ear files are intended to contain complete enterprise applications.

URI 和 URL?

URIs identify and URLs locate (on network); however, locators are also identifiers.

So all URLs are URIs (actually not quite - see below), and all URNs are URIs - but URNs and URLs are different, so you can't say that all URIs are URLs.

DAO

3

Spring


HTML
unity
HTML

什么是 Spring?

Spring 是 Java EE 的是一个轻量级的开源框架.

使 J2EE 开发更容易

通过实现基于POJO的编程模型

Spring 的核心 design pattern 是 IOC

what are benefits of using spring?

Lightweight: Spring is lightweight when it comes to size and transparency. The basic version of spring framework is around 2MB.

Inversion of control (IOC): Loose coupling is achieved in spring using the technique Inversion of Control. The objects give their dependencies instead of creating or looking for dependent objects.

Aspect oriented (AOP): Spring supports Aspect oriented programming and enables cohesive development by separating application business logic from system services.

Container: Spring contains and manages the life cycle and configuration of application objects.

MVC Framework

Transaction Management: Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

Exception Handling: Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.

Spring 都有哪些模块?

- The Core container module
- O/R mapping module (Object/Relational)
- DAO module
- Application context module
- Aspect Oriented Programming
- Web module
- MVC module

什么是 Spring 的配置文件?

Spring 的配置文件是一个 XML 文件.

这个文件包含类的或者说 bean 的信息以及它们是如何配置的

什么是依赖注入 - Dependency Injection?

反转控制 Inversion of Control (IoC) 或者叫依赖注入 Dependency Injection

is a general concept, and it can be expressed in many different ways and Dependency Injection is merely one concrete example of Inversion of Control.

IoC container 是什么?

管理 bean 的生命周期 (从创建,配置等等再到摧毁)

通过 dependency injection (DI) 管理构成一个应用各个部件

什么是 Spring beans?

一个 bean 是被实例化,组装,以及由Spring IoC容器管理的对象.

什么是自动装配

Spring 自动解决 bean 之间的关系.通过检查 BeanFactory 中的内容，而无需使用`<constructor-arg>`和`<property>`元素

什么是 AOP?

面向方面的编程 (Aspect-oriented programming),

它可以运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术. 比如检测某个模块的运行时间. 加入额外的功能 (introduce)

下面是专业的所法 :

利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率

主要的功能是：日志记录，性能统计，安全控制，事务处理，异常处理等等。

@Autowired @Inject @Resource

@Autowired and @Inject

```
Matches by Type
Restricts by Qualifiers
Matches by Name

AutowiredAnnotationBeanPostProcessor
```

@Resource

```
Matches by Name
Matches by Type
Restricts by Qualifiers

CommonAnnotationBeanPostProcessor
```

4

Hibernate


unity

get and load

get vs load is one of the most frequently asked Hibernate Interview question, since correct understanding of both get() and load() is require to effectively using Hibernate. Main difference between get and load is that, get will hit the database if object is not found in the cache and returned completely initialized object, which may involve several database call while load() method can return proxy, if object is not found in cache and only hit database if any method other than getId() is called. This can save lot of performance in some cases. You can also see difference between get and load in Hibernate for more differences and detailed discussion on this question.

Read more: http://javarevisited.blogspot.com/2013/05/10-hibernate-interview-questions-answers-java-j2ee-senior.html#ixzz3BJkBQ5vg

What is SessionFactory in Hibernate?

SessionFactory 是一个创建 hibernate Session 对象的工厂.

它可以作为单一的 data store 及也是线程安全的， 使多个线程可以使用相同的 SessionFactory.

一个 Java JEE 应用只有一个 SessionFactory 如果只有一个数据库的话

当创建之后关于 Object/Relational mapping 的元数据是不能改的.

sorted 和 ordered collection

sorted collection是在内存中通过 java 比较器进行排序的 ordered collection是在数据库中通过 order by 进行排序的

建议使用 ordered collection 避免 OutOfMemoryError 问题.

What is the file extension used for hibernate mapping file?

filename.hbm.xml

hibernate 的三种状态

临时状态(Transient):

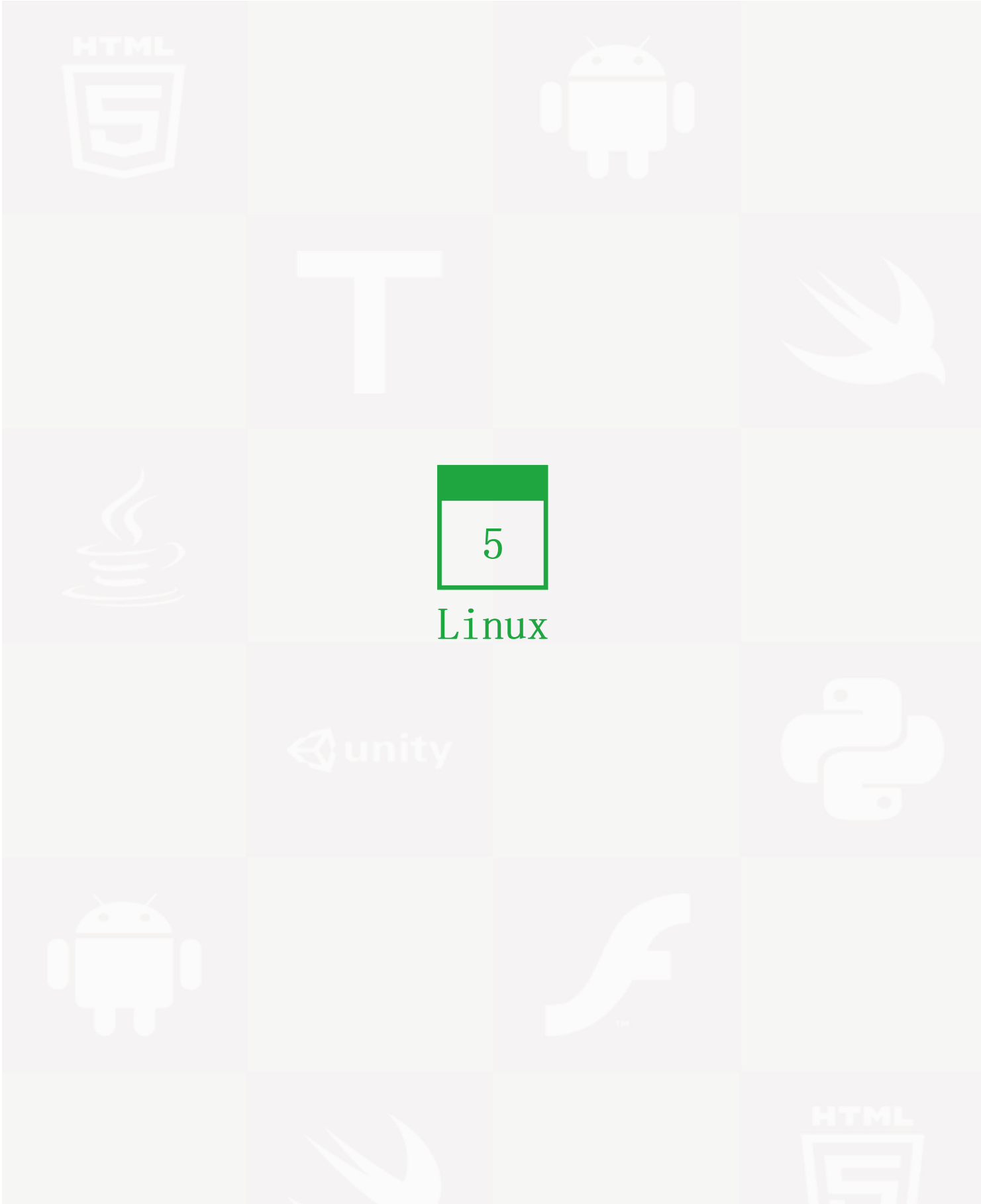
当new一个实体对象后，这个对象处于临时状态，即这个对象只是一个保存临时数据的内存区域，如果没有变量引用这个对象，则会被jre垃圾回收机制回收. 这个对象所保存的数据与数据库没有任何关系，除非通过Session的save或者SaveOrUpdate把临时对象与数据库关联，并把数据插入或者更新到数据库，这个对象才转换为持久对象.

持久状态(Persistent):

持久化对象的实例在数据库中有对应的记录，并拥有一个持久化表示（ID）. 对持久化对象进行delete操作后，数据库中对应的记录将被删除，那么持久化对象与数据库记录不再存在对应关系，持久化对象变成临时状态. 持久化对象被修改变更后，不会马上同步到数据库，直到数据库事务提交. 在同步之前，持久化对象是脏的（Dirty）.

游离状态(Detached)：

当Session进行了Close、Clear或者evict后，持久化对象虽然拥有持久化标识符和与数据库对应记录一致的值，但是因为会话已经消失，对象不在持久化管理之内，所以处于游离. 游离状态的对象与临时状态对象是十分相似的，只是它还含有持久化标识.



5

Linux


HTML
unity
HTML

查找文件

通过名字查找文件

```
find -name "nameFile"
```

但是分大小写.

如果不区分大小写用下面的命令:

```
find -iname "nameFile"
```

列出文件列表

```
ls
```

列出当前目录的所有内容

SQL

数据库管理系统在写入/更新资料过程中为保证事务是正确可靠的，其必须具备 ACID 特性。

ACID

- A: atomicity(原子性) - 一个事务（transaction）中的所有操作，要么全部完成，要么全部不完成，不会结束在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。
- C: consistency(一致性) - 在事务开始之前和事务结束以后，数据库的完整性没有被破坏。即从一个一致性状态转换到另一个一致性状态。
- I: isolation(隔离性) - 当两个或者多个事务并发访问（此处访问指查询和修改的操作）数据库的同一数据时所表现出的相互关系。通常来说，一个事务所做的修改在最终提交以前，对其他事务是不可见的。
- D: durability(持久性) - 在事务完成以后，该事务对数据库所作的更改便持久地保存在数据库之中，并且是完全的。

符合 ACID 的典型例子有银行转账，整个过程称为一个事务，具有 ACID 特性。

Reference

- ACID - 维基百科，自由的百科全书
- 《高性能 MySQL》

设计一对一

一对一可以两个实体设计在一个数据库中.例如设计一个夫妻表，里面放丈夫和妻子

One to multi

一对多可以建两张表，将一这一方的主键作为多那一方的外键，例如一个学生表可以加一个字段指向班级(班级与学生一对多的关系)

multi to multi

多对多可以多加一张中间表，将另外两个表的主键放到这个表中（如教师和学生就是多对多的关系）

都使用过哪些join?

初学者说这2个

- inner join
- (left/right) outer join

再牛逼点补充下下面2个

- cross join
- self-join

合并

How can you combine two tables/views together? For instance one table contains 100 rows and the other one contains 200 rows, have exactly the same fields and you want to show a query with all data (300 rows).

```
SELECT column_name FROM table_name1
UNION
SELECT column_name FROM table_name2
```

注意

- UNION 内部的 SELECT 语句必须拥有相同数量的列.
- 列也必须拥有相似的数据类型.
- 每条 SELECT 语句中的列的顺序必须相同.

Where 和 Having

WHERE 从句一般是在行的层级去筛选数据 (before grouping). HAVING 从句一般在 GROUP BY 之后所以是在 "groups" 的基础上删选.

更准确的说在 SQL 中增加 HAVING 子句原因是 WHERE 关键字无法与合计函数一起使用

因为在查询过程中聚合语句(sum,min,max,avg,count)要比having子句优先执行.而where子句在查询过程中执行优先级别优先于聚合语句(sum,min,max,avg,count)

What type of wildcards have you used?

首先说下什么是 wildcards

Wildcards are special characters that allow matching string without having exact match.

SQL 通配符必须与 LIKE 运算符一起使用

% 替代一个或多个字符

_ 仅替代一个字符

[charlist] 字符列中的任何单一字符

[^charlist] 或者 [!charlist] 不在字符列中的任何单一字符

7

Scrum

- 迭代式开发 Le développement itératif
- 以人为核心

为什么说是以人为核心？

我们大部分人都学过瀑布开发模型，它是以文档为驱动的，为什么呢？因为在瀑布的整个开发过程中，要写大量的文档，把需求文档写出来后，开发人员都是根据文档进行开发的，一切以文档为依据；而敏捷开发它只写有必要的文档，或尽量少写文档，敏捷开发注重的是人与人之间，面对面的交流，所以它强调以人为核心。

什么是迭代？

迭代是指把一个复杂且开发周期很长的开发任务，分解为很多小周期可完成的任务，这样的一个周期就是一次迭代的过程；同时每一次迭代都可以生产或开发出一个可以交付的软件产品。

Scrum 中的三大角色

- 产品负责人（Product Owner）

主要负责确定产品的功能和达到要求的标准，指定软件的发布日期和交付的内容，同时有权力接受或拒绝开发团队的工作成果。

- 流程管理员（Scrum Master）

主要负责整个Scrum流程在项目中的顺利实施和进行，以及清除挡在客户和开发工作之间的沟通障碍，使得客户可以直接驱动开发。

- 开发团队（Scrum Team）

主要负责软件产品在Scrum规定流程下进行开发工作，人数控制在5~10人左右，每个成员可能负责不同的技术方面，但要求每成员必须要有很强的自我管理能力，同时具有一定的表达能力；成员可以采用任何工作方式，只要能达到Sprint的目标。

What’s sprint?

Sprint是短距离赛跑的意思，这里面指的是一次迭代，而一次迭代的周期是1个月时间（即4个星期），也就是我们要把一次迭代的开发内容以最快的速度完成它，这个过程我们称它为Sprint。

How to scrum

1、我们首先需要确定一个Product Backlog，这个是由Product Owner 负责的

2、Scrum Team根据Product Backlog列表，做工作量的预估和安排

3、有了Product Backlog列表，我们需要通过 Sprint Planning Meeting 来从中挑选出一个Story作为本次迭代完成的目标，然后把这个Story进行细化，形成一个Sprint Backlog；

4、每个成员根据Sprint Backlog再细化成更小的任务（细到每个任务的工作量在2天内能完成）

5、要进行 Daily Scrum Meeting（每日站立会议），每次会议控制在15分钟左右，每个人都必须发言，并且要向所有成员当面汇报你昨天完成了什么，并且向所有成员承诺你今天要完成什么，同时遇到不能解决的问题也可以提出，每个人回答完成后，要走到黑板前更新自己的 Sprint burn down（Sprint燃尽图）；

6、做到每日集成，也就是每天都要有一个可以成功编译、并且可以演示的版本；很多人可能还没有用过自动化的每日集成，其实TFS就有这个功能，它可以支持每次有成员进行签入操作的时候，在服务器上自动获取最新版本，然后在服务器中编译，如果通过则马上再执行单元测试代码，如果也全部通过，则将该版本发布，这时一次正式的签入操作才保存到TFS中，中间有任何失败，都会用邮件通知项目管理人员；

7、当一个Story完成，也就是Sprint Backlog被完成，也就表示一次Sprint完成，这时，我们要进行 Srpint Review Meeting（演示会议），也称为评审会议，产品负责人和客户都要参加（最好本公司老板也参加），每一个Scrum Team的成员都要向他们演示自己完成的软件产品

8、最后就是 Sprint Retrospective Meeting（回顾会议），也称为总结会议，以轮流发言方式进行，每个人都要发言，总结并讨论改进的地方，放入下一轮Sprint的产品需求中；

8

Continuous integration

un ensemble de pratiques utilisées en génie logiciel consistent à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application dével oppée.

Pour appliquer cette technique, il faut d'abord que :

le code source soit partagé les développeurs commit quotidiennement des tests d'intégration soient développés pour valider l'application il faut un outil d'intégration continue

好处:

le test immédiat des unités modifiées la prévention rapide en cas de code incompatible ou manquant les problèmes d'intégration sont détectés et réparés de façon continue, évitant les problèmes de de rnière minute une version est toujours disponible pour un test, une démonstration ou une distributi on

statement 和 prepared statement?

Statement每次执行sql语句,数据库都要执行sql语句的编译. 最好用于仅执行一次查询并返回结果的情形，效率高于PreparedStatement.

Prepared statements offer better performance, as they are **pre-compiled**. Use when you plan to use the SQL statements **many times**.

Prepared statements are more secure because they use bind variables, which can prevent SQL injection attack.

Stored Procedure?

A stored procedure is a group of SQL statements that form a logical unit and perform a particular task. (粗俗的可以理解为一个定义好的方法, 提供输入就会得到对应的输出)

```
CallableStatement cs = con.prepareCall("{call MY_SAMPLE_STORED_PROC}");
ResultSet rs = cs.executeQuery();
```

What does the Class.forName("MyClass") do?

Loads the class MyClass.

Execute any static block code of MyClass.

Returns an instance of MyClass.

Connection Pooling ?

Connection Pooling is a technique used for reuse of physical connections and reduced overhead for your application.

Connection pooling functionality minimizes expensive operations in the creation and closing of sessions.

Database vendor's help multiple clients to share a cached set of connection objects that provides access to a database. Clients need not create a new connection every time to interact with the database.

What are the steps in the JDBC connection?

- Step 1 : Register the database driver by using : Class.forName(\" driver classs for that specific database\");
- Step 2 : Now create a database connection using :

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

- Step 3: Now Create a query using :

```
Statement stmt = Connection.Statement(\"select * from TABLE NAME\");
```

- Step 4 : Exceute the query :

```
stmt.exceuteUpdate();
```

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台

更多信息请访问

http://wiki.jikexueyuan.com/project/java-interview-question/