面试专题 Day01

2011/8/09

系列班级：SD1104	主讲：梁建全	助教：马茶

Day01

(一) Corejava

1. Java 基本概念

1） 面向对象的基本特征

1)抽象：抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，暂时不用部分细节。抽象包括两个方面，一是过程抽象，二是数据抽象。

2)继承：继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类（子类），而原始类称为新类的基类（父类）。 派生类可以从它的基类那里继承方法和实例变量，并且类可以修改或增加新的方法使之更适合特殊的需要。

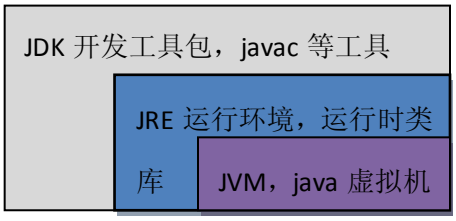
3)封装：封装是把过程和数据包围起来，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。
封装的级别由访问修饰符的不同而有所差别，执行最好封装的是 private，访问控制级别表：

	private	default	protected	public
同一个类中	√	√	√	√
同一个包中		√	√	√
子类中			√	√
全局范围内				√

4)多态性：多态性是指允许不同类的对象对同一消息作出响应。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、

代码共享的优势，很好的解决了应用程序函数同名问题。

2） JDK JRE JVM



3） GC 垃圾回收机制

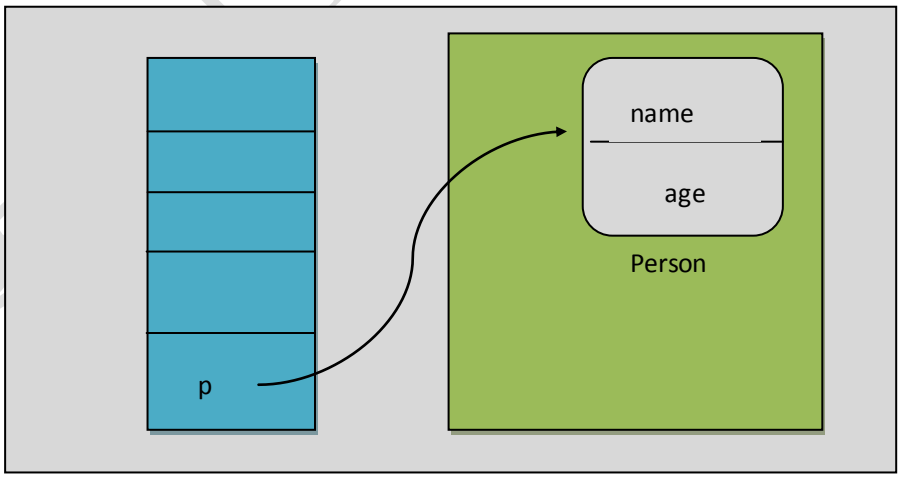
由 JVM 提供的功能，帮助程序员回收不再使用的资源，它是一个独立的线程，会定期检查虚拟机的内存状态，它是不受程序员精确控制，它是由底层系统状态来确定的。但是可以做一些工作，加快资源的释放，例如：

A、调用 System.gc();方法（不推荐）。

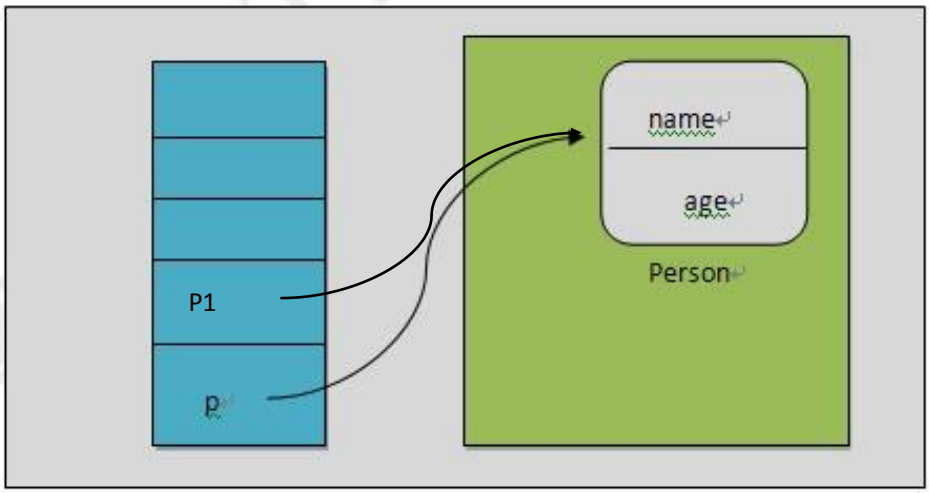
B、如果是有大量占用资源的对象，可以将对象设置为 null，加快回收（推荐使用）。

```
Person p=new Person();
```

内存状态图

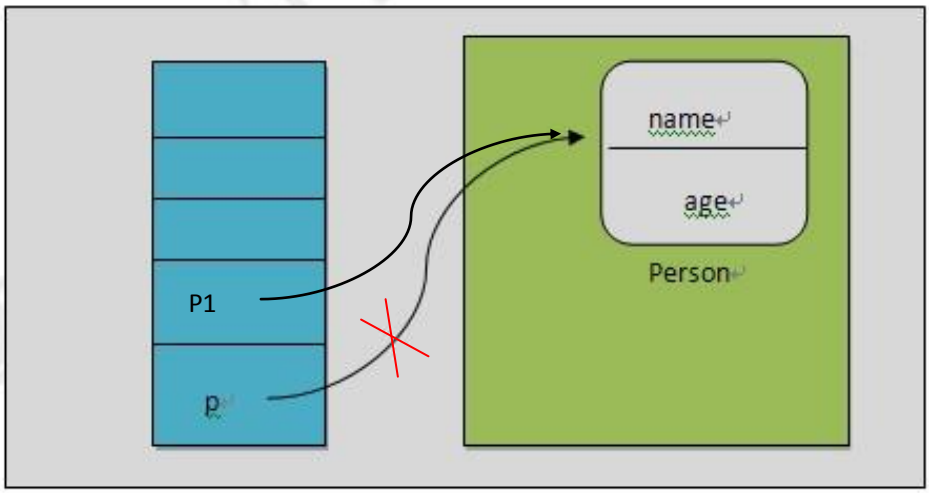



```
String s=new String();// 创建了几个对象(2 个)
Person p1=p;
```



p=null;

内存状态图：



此时内存中的对象不会被回收，因为 p1 这个引用还在使用这个对象。

4） 重写重载的区别：

重写：当父类的方法不能满足子类的需求的时候，需要对父类中的方法进行重写。 规则： 方法名、方法参数相同，返回类型可以是子类型（5.0 新特性），抛出异常不能变大，访问范围不能变小

重载：方法签名不同，方法名相同，方便程序调用者使用。

构造方法不能被重写，可以被重载

5） 抽象类、接口

抽象类定义：public abstract XXX{}，使用它有两个目的：

1.不能实例化，如果定义了一个类，不想“new”，而是想让其他的类继承；

2.该类中可以存在没有实现的方法。

接口：是一种特殊的抽象类，里面没有实现的方法，在接口中，只能包含方法的声明和常量的定义。

```
public interface c{
 int c=2;//默认为 public static final 类型的，常量
}
```

当类中的所有方法只是声明的时候，使用接口。方法有部分实现使用抽象类。

2. java 关键字、运算符、基本语法：

1）goto、const 是否为关键字？是 java 中的保留关键字

2）static、final、this、super

static：静态的，可以修饰属性、方法、语句块。方法提倡使用“类名.”直接调用。静态的语句块在类加载的时候只执行一次。

final:修饰属性（不能被修改）、类（不能被继承）、方法（不能被重写）、

试题 1.String 类可不可以被继承？

不可以，应为它是 final 类.

试题 2.private final Person p=new Person(“tom”,22);

p.setName(“jack”);//有没有错误？

没有错误,Final 修饰应用类型变量，里面的内容只是可以变得，因为引用没有变化。

试题 3.Final finalize finally 的区别？

This：代表当前类型，在 getter，setter 方法上使用,解决冲突问题。

```
public void setAge(int age){
 this.age=age;
 }
```

在构造方法中使用：

```
public Person(){
 This(name ,age);
}
```

Super：调用父类的构造。

This 和 super 不可以使用在有 static 修饰的语句块和方法中，应为 this 和

super 是非静态的对象类型，而 static 中只可以调用静态的变量。

3）& 和 &&，|和||的区别：

&&是短路与，效率比&高。只要有一个 false，&&便返回 false，不再对后面的表达式做运算，而&还会对其他的表达式进行计算。

例：F 类

4）运算符：

```
short s=9;
s=s+1;//是否会编译通过（有错误，int 类型的值付给 short 类型，会出错）
s+=1；//是否会编译通过（+=在过程中执行了强制类型转换）
System.out.println(12/5);//输出的值为？（2）
System.out.println(12/5.0);//输出的值为？（2.4）
```

自增“++“，自减”–“运算符：

```
class c{
Static{
  Int i=3;
 }
 Static int i;
 Static int j;
 Public static void main(String[] args){
  j=i++;
  System.out.println(i+++j++);//输出结果为？
 }
}
```

5）分支语句

```
Switch（S）//s 可以为：byte short int char 枚举（使用的较少）
                //五种类型,在 5.0 之后（自动装箱、拆箱），对应的封转
                //类型也可以使用。
```

试题：exc1 第 3 题

3. Java 数据类型，集合等 API

1)数据类型：基本类型、引用类型两种

基本类型：byte、short、int、long、double、float、char、boolean

对应封装类型：Byte、Short、Integer、Long、Double、Float、Char、Boolean

JDK1.5 之后添加自动拆箱装箱的功能，可以实现基本类型和封装类的自动转化

笔试题集 8

试题：基本类型不是很好使用吗？直接操作栈中的内容，效率很高，为什么还要有封装类的出现呢？有什么意义?

封转类可以使用集合来存储

封装类是 Object 的子类，可以使用 Object 中的方法。

封装类的本身提供了一些方便使用的 API。

2)重要 API

1. String 类型的重点方法：

String s=”abc”;与 String s=new String(“abc”);的区别。

示例代码：G 类

String s=”abc”;//会在缓冲池中查找有没有字符串 abc，如果有的话直接拿过来使用，没有的话创建对象。

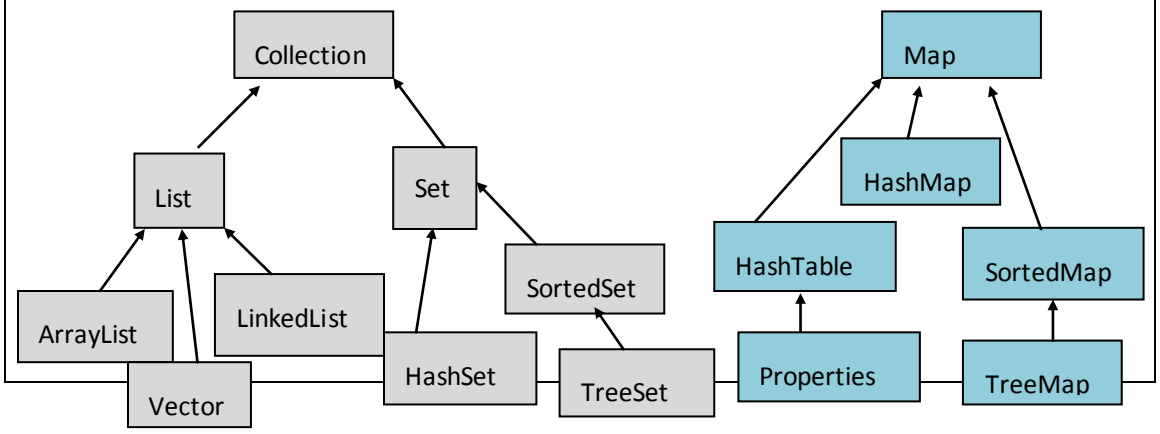
2. Object 类中的重要方法：

Equals()与“==“比较

3. StringBuffer、StringBuilder、String 的区别?

String 是不可改变的字符序列，StringBuffer、StringBuilder 是可改变的。StringBuffer 是线程安全的，StringBuilder 是非线程安全的，它的效率高。

3) 集合



1. List 和 set 集合的区别：

List：有序，可重复，set：无序，不可重复

2. List 子类作对比：

ArrayList：线性表，以数组实现，查询的效率高。

LinkedList：使用双向链表实现，增删改的效率高。

Vector：底层也是使用数组实现，是线程安全的。、

3. Set 是一种无序不可重复（指的是不是按照放入顺序存储的），TreeSet 是可排序的，默认按自然顺序排列，实现 comparator 和 comparatable。取值通过迭代来实现。

4. Map：不继承自 collection，是以键值对存储的，Key-value。

Collections/Collection 的区别：

HashMap 和 HashTable 的区别：HashMap 可以使用 null 值，非同步的集合 HashTable 是线程安全的。

4) io 流

分类	字节输入流	字节输出流	字符输入流	字符输出流
抽象基类	InputStream	OutputStream	Reader	Writer
访问文件	FileInputStream	FileOutputStream	FileReader	FileWriter
访问数组	ByteArrayInputstream	ByteArrayOutPutStream	CharArrayReader	CharArrayWriter
访问字符串			StringReader	StringWriter
缓冲流	BufferedInputStream	BufferedOutputStream	BufferedReader	BufferedWriter
转换流			InputStreamReader	OutputStreamWriter
对象流	ObjectInputStream	ObjectOutputStream		
打印流		PrintStream		PrintWriter
特殊流	DataInputStream	DataOutputStream		

字节流和字符流的区别：

字符流与字节流的桥梁：

序列化接口：作用

如何序列化：1,实现 Serializable，按照默认机制，没有特殊需要。也可以强制重写方法。

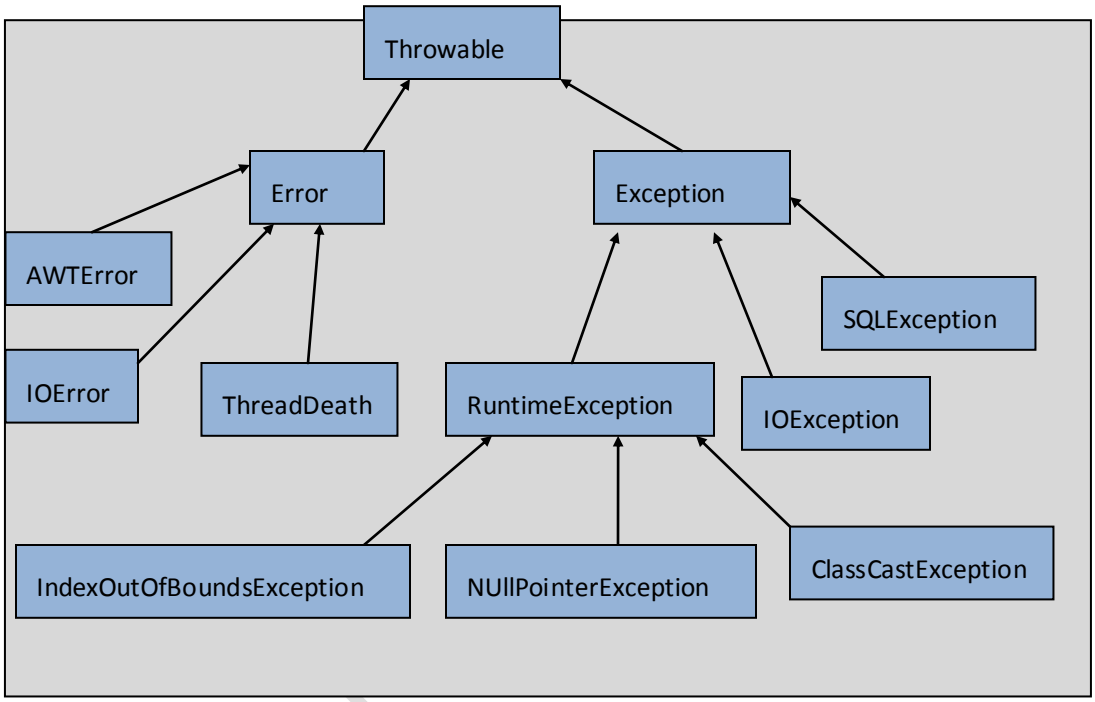
2.实现 Externalizable，有两个方法需要做

（writerExternal 和 readExternal 方法），强制自定义序列化.

6） Math 类提供的基本方法：见 I 类

```
Math.round(11.5);
Math.round(-11.5);//ceil() floor()
```

4. java 异常处理



异常处理机制

Java 异常处理机制主要依赖于 Try、catch、finally、throws、throw 五个关键字，其中 try 关键字后紧更一对大括号，简称 try 块。它里面可放置可能引发异常的代码。Catch 后是对应异常类型和一个代码块，可以跟有多个 catch 块，catch 块后面可以跟一个 finally，它用于回收资源。Throws 关键字主要用在方法签名中，而 throw 则是可以单独使用的，抛出一个异常对象。

试题 1：何时使用 try catch 处理，何时使用 throws？

如果异常自己可以处理，使用 try catch 来捕获异常，若是自己无法处理异常，则使用 throws 抛出，交给方法调用者来处理。另外使用 try catch 对程序有一定的修复作用。

自定义异常：继承 Exception 或 RuntimeException，自定义异常还需要提供两种构造器，一个无参的，一个是带一个字符串的构造器，字符串是对该异常对象的

详细说明。

Day02

5. 线程

1) 如何创建线程

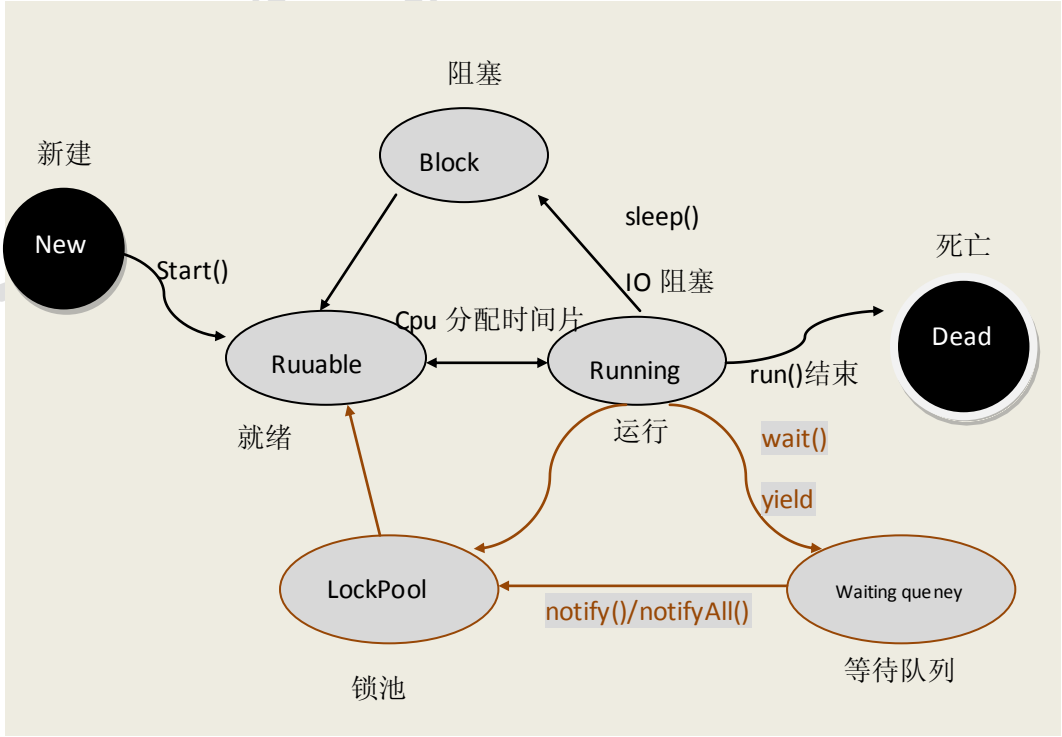
A、继承 Thread 类

```
class   A extends Thread{
    public void run (){
     }
}
```

B、 实现 Runnable 接口

```
class B implements Runnable{
  public void run(){
  }
}
```

2) 线程的状态



3) 线程同步，通信

A、 数据共享：

有三个线程：t1 t2 t3 临界资源。多个线程同时对一个数据进行修改

```
class A extends Thread{
  static int i=1;//i 为临界资源， 多个线程共享一个 i 变量
  public void run(){
     try{
      Thread.sleep(1000);// 将线程的并发性的几率提高
     }catch(Exception e){
     }
     i++;
     System.out.println("i="+i);
     }
  }
```

B、保护临界资源，使用 synchronized 关键字

```
   class A extends Thread{
  static Integer count=1;//i 为临界资源，多个线程共享一个 i 变量
  public void run(){
     try{
      Thread.sleep(1000);// 将线程的并发性的几率提高
     }catch(Exception e){
     }
    synchronized(count){//加黑部分为同步代码块，解决并发性
     count++;
     System.out.println("count="+count);
     }
     }
  }
```

也可以在方法前面使用 synchronized 修饰，相当于使用 this 当前对象的锁来实现

锁机制，此时同步还是没有实现，因为他们各自拿着各自的锁。

在使用 synchronized 语句块的使用过程中会遇到 wait()和 notify()方法的使用。

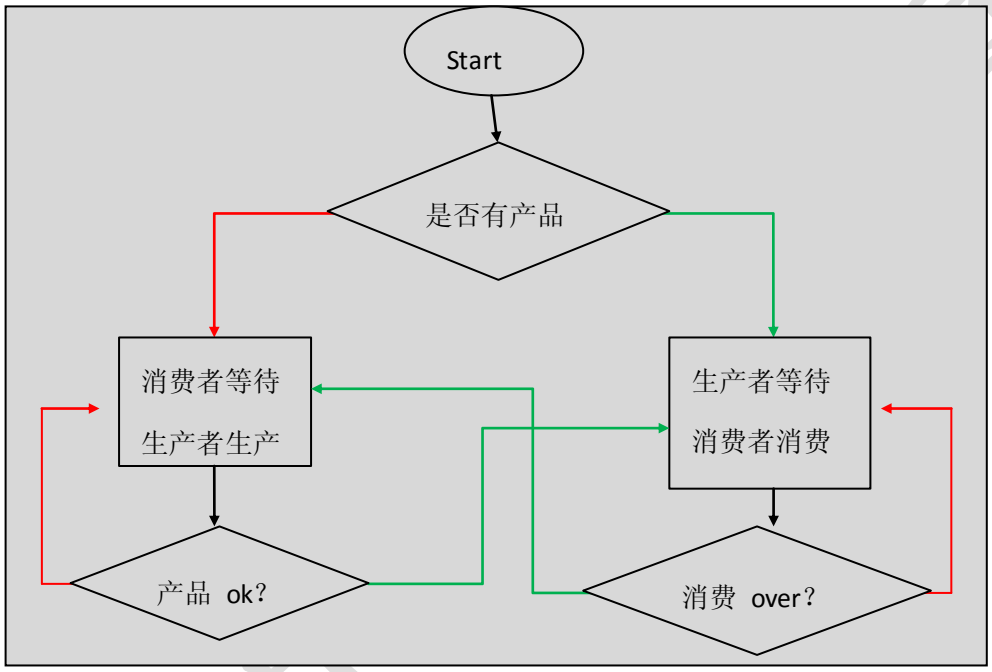
4) Lock 实现锁机制（JDK5.0 之后提供）示例：thread 项目

java.util.concurrent.locks.ReentrantLock 类

lock.lock();//使当前线程持有锁

lock.unLock();//使当前线程释放锁

5) 以下是消费者与生产者图示：



（二） web 部分

1. 什么是 servlet？它的生命周期？

Servlet 通常被称作为服务器端小程序，用于处理及响应客户端的请求。它是一个特殊的 java 类，必须继承自 HttpServlet。

Servlet 生命周期：

A. 创建 Servlet 实例。有两个时机:

1)客户端第一次请求某个 Servlet 时，servlet 容器创建 servlet 实例；

2)web 应用启动时立即创建 servlet 实例，即 load-on-startup servlet。

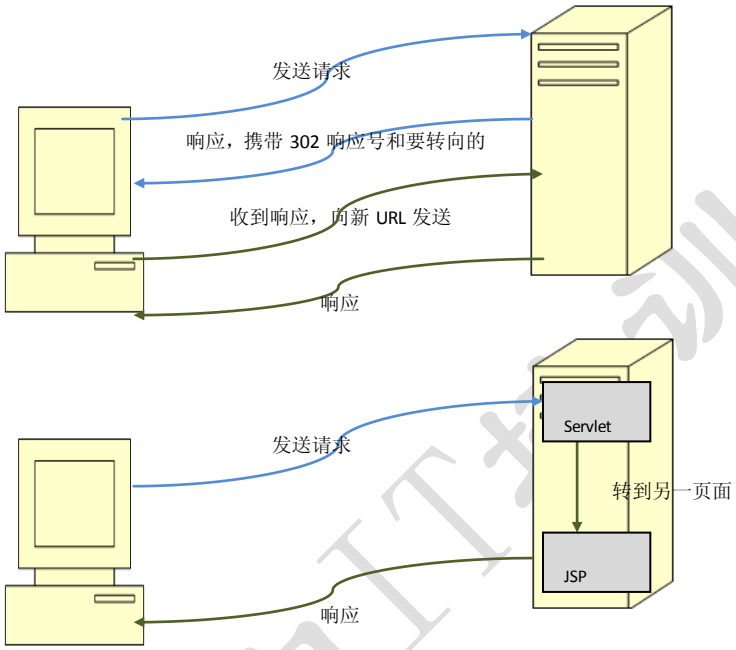
B. Web 容器调用 servlet 的 init 方法，对 servlet 进行初始化。

C．Servlet 初始化之后，将一直处于容器中，用于响应客户端请求，如果客户发 get 请求，容器调用 doGet 方法来处理请求；如果客户发送 POST 请求，容器调用 doPost 方法来处理请求。或者统一调用 service 方法来处理请求。

D．Web 容器决定销毁 servlet 实例时，会调用 servlet 的 destory 方法，通常是在关闭 web 应用时，销毁 servlet。

2. forward 和 redirect 的区别?（处理机制，响应机制）

两者响应如下图:



区别：

A、forward 只能转发给同一个应用中的组件，地址以“/“开头，代表的是当前 WEB 应用程序的根目录；

redirect 既可以是同应用中的，也可以是其他应用中的，地址以”/“开头，代表的是整个 WEB 站点的根目录。

B、forward 转发之后浏览器的地址栏不会有变化，应为转发是在服务器端进行的，浏览器并不知道服务器有着一个动作；

Redirect 重定向之后浏览器的地址会发生变化。

C、forward 转发只给服务器发送了一个请求；

Redirect 是给服务器发送了两个请求。

D、forword 的调用者和被调用者之间使用的是同一个 Request 对象；

Redirect 的调用者和被调用者之间使用的是两个不同的 request 对象。

3. post 与 get 的区别：

A、数据传输的方式不同：get 借助于协议头，post 借助于协议体

B、数据的传输量不同：get 传输量受 url 限制，post 不受限制

C、传输的数据格式：get 只能提交字符串格式的，post 可以传输对象类型。

D、编码方式不同：get 提交的编码是 url 地址的编码 ISO-8859-1，post 可以的编码与 jsp 页面指定的一样。

E、安全性

4. 常用的 JSP 内置对象有哪些？

request：HttpServletRequest 的实例，封装了一次请求信息，客户端的请求都被封装在该对象里。常用的方法有：getParameter(String name), getParameterValues(String name),setAttribute(String name,Object value),getAttribute(String name)和 setCharacterEncoding(String enc)等。

response：HttpServletResponse 的实例，代表服务器对客户端的响应，response 对象通常用于重定向，常用的方法有 getOutputStream(),sendRedirect() 等。

session：HttpSession 的实例，代表一次会话，当客户端浏览器与站点建立连接时，会话开始；当客户端关闭浏览器时，会话结束。常用的方法有：getAttribute(String name),setAttribute(String name,Object value)等。

application：ServletContext 对象,该实例代表 JSP 所属的 web 应用本身，可用于 jsp 页面，或者在 Servlet 之间交换信息。

out：PrintWriter 的实例，它代表 JSP 页面的输出流，用于输出内容。

Exception：java.lang.Throwable 的实例，该实例代表其他页面中的异常和错误。

Page：代表该页面本身。

pageConext:PageContext，代表 JSP 页面上下文。

config:ServletConfig 实例，代表 JSP 页面的配置信息。

5. JSP 中的动态引入，和静态引入

区别：

A、 静态代码是将被导入页面的代码完全导入，两个页面融合成一个 servlet；而动

态导入则在 servlet 中实用 include 方法来引入被导入的页面内容。

B、 静态导入时被导入页面的编译指令会起作用；而动态导入时被导入页面的编译指令则失去作用，只是插入被导入页面的 body 内容。

6.page、request、session、application、cookie 的使用范围及使用经验

Page:同一个页面中共享信息。

Request：同一个请求中共享信息。

Session:同一用户不同界面。

Application：不同用户，不同页面。

Cookie：存储在客户端，存储不重要的不敏感的信息。

页面之间传值

7．描述 MVC 模式

MVC 如何划分，各部分的职责和优点

MVC 模式将一个应用分成 3 个基本部分：Model、View、Controller，这三部分以最少的耦合协同工作，提高应用的可扩展性和可维护性。

MVC 有如下优点：

1. 多个视图可以共享一个模型。
2. 模型返回的数据与现实逻辑分离。
3. 应用被分割为三层，降低了各层之间的耦合，提供了应用的可扩展性。

侧重改善程序结构，便于修改和维护，

8. 描述 struts1 与 struts2 的区别，常用类和处理流程

A. struts1 的常用 API：ActionForm(封装表单信息),ActionServlet(控制器),RequestProcessor(处理请求信息),Action,ActionForward,ActionMapping

Struts1 流程：

(1)Tomcat 服务器启动时,根据 web.xml 配置,创建 ActionServlet 对象,并解析 struts-config.xml 配置文件

(2)客户端发送*.do 请求,请求进入 ActionServlet,它会根据 struts-config.xml 中的配置,找相应的<action>元素配置处理

(3)ActionServlet 获取请求 uri,然后与<action>元素的 path 属性匹配,匹配成功将按照<action>元素执行处理

(4)ActionServlet 根据<action>元素的 name 属性创建 ActionForm 对象,接收请求信息.然后将其存入 scope 指定范围,key 值为 attribute 属性

(5)ActionServlet 根据<action>元素的 type 属性创建 Action 对象,执行 execute()方法.并将 ActionForm 对象当参数传入

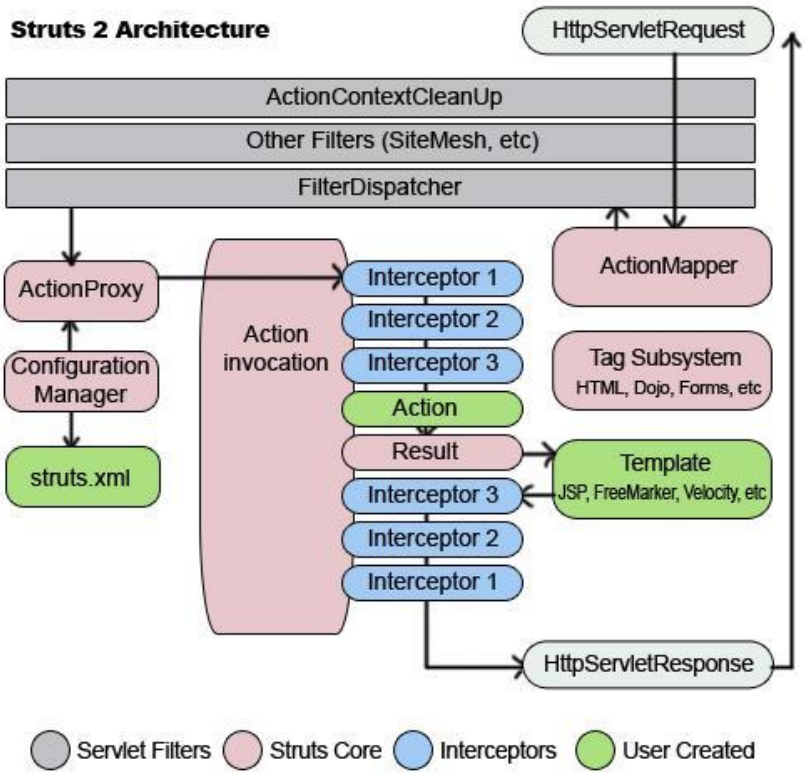
(6)调用 Action 的 execute 方法之后,

该方法返回一个 ActionForward 对象给 ActionServlet

(7)ActionServlet 根据 ActionForward 对象信息调用 JSP 视图做响应.

B. Struts2 常用 API：StrutsPrepareAndExecuteFilter，ActionContext、ServletActionContext、ValueStack、 Result、Interceptor

Struts2 流程：



9. Hibernate 的好处:

对 JDBC 进行了封装，简化了数据库访问操作。

- 对象/关系数据库映射
- 透明持久化，当对象与 session 相关联，对对象的操作就会直接反应到数据库的操作，直到此对象与 session 失去关联。

- 它没有入侵性，是轻量级框架。
- 移植性很好
- 缓存机制，提供一级缓存、二级缓存和查询缓存

10. Hibernate 是如何延迟加载的？

延迟加载：

延迟加载机制是为了避免一些无谓的性能开销而提出来的，所谓延迟加载就是当在真正需要数据的时候，才真正执行数据加载操作。在 Hibernate 中提供了对实体对象的延迟加载以及对集合的延迟加载，另外在 Hibernate3 中还提供了对属性的延迟加载 ，有三种延迟加载方式：

A、实体对象的延迟加载：

如果想对实体对象使用延迟加载，必须要在实体的映射配置文件中进行相应的配置，通过将 dass 的 lazy 属性设置为 true，来开启实体的延迟加载特性。

B、集合类型的延迟加载：

对于集合类型的延迟加载意义重大，因为这有可能使性能得到大幅度的提高，通过使用自定义集合类的实现 。通过将<set>元素的 lazy 属性设置为 true 来开启集合类型的延迟加载特性。

C、属性延迟加载：

这个机制又为获取高性能查询提供了有力的工具。我们可以通过属性延迟加载机制，来使我们获得只有当我们真正需要操作这个字段时，才去读取这个字段数据的能力，通过对<property>元素的 lazy 属性设置 true 来开启属性的延迟加载。

11. hibernate 的缓存

一级缓存：默认启用，当通过 session 在对对象进行增删改查操作时，如果 session 中还不存在该对象，hibernate 会将该对象放入到一级缓存中。当清理缓存时，hibernate 会将这些对象的状态同步到数据库。 Session 中的两个方法可以用来清理缓存：evict(Object obj)清理指定的持久化对象；dear()清空缓存中的所有持久化对象。

二级缓存：它是基于 sessionFactory 来实现，启动二级缓存需要增加配置内容，二级缓存启动之后，执行数据库操作用，hibernate 会先从一级缓存中查找又没有需要的数据，没有的话再去查找二级缓存，如果二级缓存中任然没有，就查询数据库。一些经常修改，重要的数据不要使用二级缓存来存储。

查询缓存：它的使用基于二级缓存，主要是将经常执行查询，而不经常修改的数据存入二级缓存，在第二次试用的时候提高效率。

12. 如何优化 hibernate

- 使用批量的增删改，可以使用原生的 sql 来提高性能
- 应用缓存来提高性能
- 结合实际情况选用正确的抓取策略和抓取时机
- 表与表之间的关系不要太复杂

  详细参看 hibernate 帮助文档-提升性能

13. 为什么使用 Spring

Spring 是一个轻量级框架，它完成了开发中的大部分的通用步骤，留给开发者的只是与特定应用相关的部分，从而大大提高了企业应用开发的效率。总结起来它有以下优点：

A. 低入侵时设计，代码的污染低。

B. Spring 中的 DI 容器降低了业务对象替换的复杂性，提高了组件之间的解耦。

C. 使用 spring 中的 AOP，允许将一些通用的功能如：安全、事务、日志等进行集中处理，从而提供了更好的复用。

D. Spring 的高度开放性，并不强制应用完全依赖于 spring，开发者可以选用 spring 的部分功能。

14. 解释一下 Dependency injection 和 IOC

DI 依赖注入，IOC 控制反转，其含义完全相同，当某个 java 实例需要调用另外一个 java 实例的时候，不再有调用者来创建被调用者的实例，创建被调用者的实例由 spring 容器来完成，然后注入给调用者。Spring 的依赖注入对调用者和被调用者没有任何的要

求，完全支持 POJO 之间依赖关系的管理。

依赖注入通常有两种方式：

- 设置注入：IOC 容器通过 setter 方法来注入被依赖的实例
- 构造注入:IOC 容器通过构造起来注入被依赖的实例

15.什么是 aop，aop 的作用

AOP（Aspect Orient Programming），面向切面编程，是对 OOP（面向对象）编程的一个补充，它从程序运行的角度考虑程序的流程，提取业务处理过程的切面，它不于特定的代码耦合，它具有个步骤之间良好隔离性和源代码无关性的特性。

AOP 一般用于实现通用的功能，如事务管理、安全检查、记录日志等等。

16.spring 事务的传播行为与隔离级别，如何管理事务（了解）

17.谈谈 ssh 整合

主要是回答 struts、spring、hibernate 整合之后各个框架的作用是什么？

Struts 在其中主要是处理请求，决定请求的转向，hibernate 用来访问数据库，spring 贯穿在两个框架之中，实例化 action 组件所依赖的对象，并且注入到需要它的属性中。

Spring 使用它提供的 AOP 来实现 hibernate 的事务管理，以及拦截器的功能。