**面试题大集合**

# String s=new String("xyz")究竟创建了几个String Object? 二者之间有什么区别?

String s=new String("xyz")究竟创建String Object分为两种情况：

1.如果String常理池中，已经创建"xyz"，则不会继续创建，此时只创建了一个对象new String("xyz")；  
2.如果String常理池中，没有创建"xyz"，则会创建两个对象，一个对象的值是"xyz"，一个对象new String("xyz")。

# **[String类和StringBuffer类的区别](http://www.cnblogs.com/rpp506815950/archive/2012/11/07/2758646.html)**

**首先，String和StringBuffer主要有2个区别：**

**（1）String类对象为不可变对象，一旦你修改了String对象的值，隐性重新创建了一个新的对象，释放原String对象，StringBuffer类对象为可修改对象，可以通过append()方法来修改值**

**（2）String类对象的性能远不如StringBuffer类。**

# **[如何把一段逗号分割的字符串转换成一个数组?](http://www.cnblogs.com/fifiyong/p/6413369.html)**

1 用正则表达式，代码大概为： String [] result = orgStr.split(“,”);  
2 用 StingTokenizer ,代码为： StringTokenizer tokener =StringTokenizer(orgStr,”,”);  
String [] result =new String[tokener .countTokens()];  
Int i=0;  
while(tokener.hasNext(){result[i++]=toker.nextToken();}

# **[多](http://www.cnblogs.com/xilichenbokeyuan/p/6382998.html)****[线程有几种实现方法？同步有几种实现方式](http://www.cnblogs.com/xilichenbokeyuan/p/6382998.html)**

多线程实现：继承Thread类，重写run()；

实现Runnable接口，重写run()；

实现Callable接口，重写call函数

同步方式：synchronized修饰,wait(),notify()

# @requestBody和@responseBody

简介：

@RequestBody

作用：

      i) 该注解用于读取Request请求的body部分数据，使用系统默认配置的HttpMessageConverter进行解析，然后把相应的数据绑定到要返回的对象上；

      ii) 再把HttpMessageConverter返回的对象数据绑定到 controller中方法的参数上。

使用时机：

A) GET、POST方式提时， 根据request header Content-Type的值来判断:

    application/x-www-form-urlencoded， 可选（即非必须，因为这种情况的数据@RequestParam, @ModelAttribute也可以处理，当然@RequestBody也能处理）；

    multipart/form-data, 不能处理（即使用@RequestBody不能处理这种格式的数据）；

    其他格式， 必须（其他格式包括application/json, application/xml等。这些格式的数据，必须使用@RequestBody来处理）；

B) PUT方式提交时， 根据request header Content-Type的值来判断:

    application/x-www-form-urlencoded， 必须；

    multipart/form-data, 不能处理；

    其他格式， 必须；

说明：request的body部分的数据编码格式由header部分的Content-Type指定；

@ResponseBody

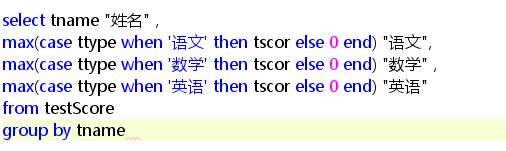
作用：

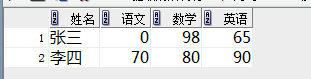
      该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区。

使用时机：

      返回的数据不是html标签的页面，而是其他某种格式的数据时（如json、xml等）使用；

# ****数据库查询案例****





# 怎样使用Ajax实现同步和异步

Jquery的async:false [这个属性默认为true，true：异步，false：同步]

# forward 和redirect的区别

1.从[地址栏](https://www.baidu.com/s?wd=%E5%9C%B0%E5%9D%80%E6%A0%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)显示来说  
forward是服务器请求资源,服务器直接访问目标地址的[URL](https://www.baidu.com/s?wd=URL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank),把那个[URL](https://www.baidu.com/s?wd=URL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)的响应内容读取过来,然后把这些内容再发给[浏览器](https://www.baidu.com/s?wd=%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank).[浏览器](https://www.baidu.com/s?wd=%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)根本不知道服务器发送的内容从哪里来的,所以它的[地址栏](https://www.baidu.com/s?wd=%E5%9C%B0%E5%9D%80%E6%A0%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)还是原来的地址.  
redirect是[服务端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)根据逻辑,发送一个状态码,告诉[浏览器](https://www.baidu.com/s?wd=%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)重新去请求那个地址.所以[地址栏](https://www.baidu.com/s?wd=%E5%9C%B0%E5%9D%80%E6%A0%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)显示的是新的[URL](https://www.baidu.com/s?wd=URL&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank).  
  
2.从[数据共享](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%85%B1%E4%BA%AB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9Pjc3P10LP16YujRkPhu90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1R1nW6snj6z" \t "https://zhidao.baidu.com/question/_blank)来说  
forward:转发页面和转发到的页面可以共享request里面的数据.  
redirect:不能共享数据.  
  
3.从运用地方来说  
forward:一般用于用户登陆的时候,根据角色转发到相应的模块.  
redirect:一般用于用户注销登陆时返回主页面和跳转到其它的网站等.  
  
4.从效率来说  
forward:高.  
redirect:低.

# 运维工程师的工作内容

在软件产品的整个生命周期中[运维工程师](https://www.baidu.com/s?wd=%E8%BF%90%E7%BB%B4%E5%B7%A5%E7%A8%8B%E5%B8%88&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)都需要适时地参与并发挥不同得作用，因此[运维工程师](https://www.baidu.com/s?wd=%E8%BF%90%E7%BB%B4%E5%B7%A5%E7%A8%8B%E5%B8%88&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)的[工作内容](https://www.baidu.com/s?wd=%E5%B7%A5%E4%BD%9C%E5%86%85%E5%AE%B9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)和方向非常多：  
　　事件管理：目标是在服务出现异常时尽可能快速的恢复服务，从而保障服务的可用性；同时深入分析故障产生 的原因，推动并修复服务存在的问题，同时设计并开发相关的预案以确保服务出现故障时可以高效的止损。在这方面主要[工作内容](https://www.baidu.com/s?wd=%E5%B7%A5%E4%BD%9C%E5%86%85%E5%AE%B9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)有：  
　　问题发现：设计并开发高效的监控平台和告警平台，使用[机器学习](https://www.baidu.com/s?wd=%E6%9C%BA%E5%99%A8%E5%AD%A6%E4%B9%A0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)、大数据分析等方法对系统中的大量监控数据进行汇总分析，以期在系统出现异常的时候可以快速的发现问题和判断故障的影响。  
　　问题处理：设计并开发高效的问题处理平台和工具，在系统出现异常的时候可以快速/自动决策并触发相关损预案，快速恢复服务。  
　　问题跟踪：通过分析问题发生时系统的各种表现(日志、变更、监控)确定问题发生的根本原因，制定并开发预

案工具。  
　　变更管理：以可控的方式，尽可能高效的完成[产品功能](https://www.baidu.com/s?wd=%E4%BA%A7%E5%93%81%E5%8A%9F%E8%83%BD&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)的迭代的变更工作。在这方面运主要[工作内容](https://www.baidu.com/s?wd=%E5%B7%A5%E4%BD%9C%E5%86%85%E5%AE%B9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)有：  
　　配置管理：通过配置管理平台(自研、开源)管理服务涉及到的多个模块、多个版本的关系以及配置的准确性。  
　　发布管理：通过构建自动化的平台确保每一次版本变更可以安全可控地发布到生产环境。  
　　容量管理：在服务运行维护阶段，为了确保服务架构部署的合理性同时掌握服务整体的冗余，需要不断评估系 统的[承载能力](https://www.baidu.com/s?wd=%E6%89%BF%E8%BD%BD%E8%83%BD%E5%8A%9B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)，并不断优化之。在这方面主要工作内容有：  
　　容量评估：通过技术手段模拟实际的用户请求，测试整个系统所能承担的最大吞吐；通过建立容量评估模型分 析[压力测试](https://www.baidu.com/s?wd=%E5%8E%8B%E5%8A%9B%E6%B5%8B%E8%AF%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)过程中的数据以评估整个服务的容量。  
　　容量优化：基于容量评估数据，判断系统的瓶颈并提供容量优化的解决方案。比如通过调整系统参数、优化服 务部署架构等方法来高效的提升系统容量。  
　　架构优化：为了支持产品的不断迭代，需要不断的进行架构优化调整。以确保整个产品能够在功能不断丰富和 复杂的条件下，同时保持[高可用性](https://www.baidu.com/s?wd=%E9%AB%98%E5%8F%AF%E7%94%A8%E6%80%A7&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uARdnvcYuHTkmHRvm1Dv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjfdrjbknWnd" \t "https://zhidao.baidu.com/question/_blank)。

# SpringMVC 和 struts2 的区别和总结

**1、**Struts2是类级别的拦截， 一个类对应一个request上下文，SpringMVC是方法级别的拦截，一个方法对应一个request上下文，而方法同时又跟一个url对应,所以说从[架构](http://lib.csdn.net/base/architecture" \o "大型网站架构知识库" \t "http://blog.csdn.net/chenleixing/article/details/_blank)本身上SpringMVC就容易实现restful url,而struts2的架构实现起来要费劲，因为Struts2中Action的一个方法可以对应一个url，而其类属性却被所有方法共享，这也就无法用注解或其他方式标识其所属方法了。

**2、**由上边原因，SpringMVC的方法之间基本上独立的，独享request response数据，请求数据通过参数获取，处理结果通过ModelMap交回给框架，方法之间不共享变量，而Struts2搞的就比较乱，虽然方法之间也是独立的，但其所有Action变量是共享的，这不会影响程序运行，却给我们编码 读程序时带来麻烦，每次来了请求就创建一个Action，一个Action对象对应一个request上下文。  
**3、**由于Struts2需要针对每个request进行封装，把request，session等servlet生命周期的变量封装成一个一个Map，供给每个Action使用，并保证线程安全，所以在原则上，是比较耗费内存的。

**4、** 拦截器实现机制上，Struts2有以自己的interceptor机制，SpringMVC用的是独立的AOP方式，这样导致Struts2的配置文件量还是比SpringMVC大。

**5、**SpringMVC的入口是servlet，而Struts2是filter（这里要指出，filter和servlet是不同的。以前认为filter是servlet的一种特殊），这就导致了二者的机制不同，这里就牵涉到servlet和filter的区别了。

**6、**SpringMVC集成了Ajax，使用非常方便，只需一个注解@ResponseBody就可以实现，然后直接返回响应文本即可，而Struts2拦截器集成了Ajax，在Action中处理时一般必须安装插件或者自己写代码集成进去，使用起来也相对不方便。

**7、**SpringMVC验证支持JSR303，处理起来相对更加灵活方便，而Struts2验证比较繁琐，感觉太烦乱。

**8、**[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/chenleixing/article/details/_blank) MVC和Spring是无缝的。从这个项目的管理和安全上也比Struts2高（当然Struts2也可以通过不同的目录结构和相关配置做到SpringMVC一样的效果，但是需要xml配置的地方不少）。

**9、** 设计思想上，Struts2更加符合OOP的编程思想， SpringMVC就比较谨慎，在servlet上扩展。

**10、**SpringMVC开发效率和性能高于Struts2。  
 **11、**SpringMVC可以认为已经100%零配置。

# Mybatis 和Hibernate 的区别

以前没怎么用过mybatis，只知道与hibernate一样是个orm数据库框架。随着使用熟练度的增加，发现它与hibernate区别是非常大的，结合至今为止的经验，总结出以下几点：

1. hibernate是全自动，而mybatis是半自动。

hibernate完全可以通过对象关系模型实现对数据库的操作，拥有完整的JavaBean对象与数据库的映射结构来自动生成sql。而mybatis仅有基本的字段映射，对象数据以及对象实际关系仍然需要通过手写sql来实现和管理。

2. hibernate数据库移植性远大于mybatis。

hibernate通过它强大的映射结构和hql语言，大大降低了对象与数据库（oracle、**[MySQL](http://lib.csdn.net/base/14" \o "undefined" \t "http://blog.csdn.net/fupengyao/article/details/_blank)**等）的耦合性，而mybatis由于需要手写sql，因此与数据库的耦合性直接取决于程序员写sql的方法，如果sql不具通用性而用了很多某数据库特性的sql语句的话，移植性也会随之降低很多，成本很高。

3. hibernate拥有完整的日志系统，mybatis则欠缺一些。

hibernate日志系统非常健全，涉及广泛，包括：sql记录、关系异常、优化警告、缓存提示、脏数据警告等；而mybatis则除了基本记录功能外，功能薄弱很多。

4. mybatis相比hibernate需要关心很多细节

hibernate配置要比mybatis复杂的多，学习成本也比mybatis高。但也正因为mybatis使用简单，才导致它要比hibernate关心很多技术细节。mybatis由于不用考虑很多细节，开发模式上与传统jdbc区别很小，因此很容易上手并开发项目，但忽略细节会导致项目前期bug较多，因而开发出相对稳定的软件很慢，而开发出软件却很快。hibernate则正好与之相反。但是如果使用hibernate很熟练的话，实际上开发效率丝毫不差于甚至超越mybatis。

5. sql直接优化上，mybatis要比hibernate方便很多

由于mybatis的sql都是写在xml里，因此优化sql比hibernate方便很多。而hibernate的sql很多都是自动生成的，无法直接维护sql；虽有hql，但功能还是不及sql强大，见到报表等变态需求时，hql也歇菜，也就是说hql是有局限的；hibernate虽然也支持原生sql，但开发模式上却与orm不同，需要转换思维，因此使用上不是非常方便。总之写sql的灵活度上hibernate不及mybatis。

总结：

mybatis：小巧、方便、高效、简单、直接、半自动

hibernate：强大、方便、高效、复杂、绕弯子、全自动

mybatis：

1. 入门简单，即学即用，提供了数据库查询的自动对象绑定功能，而且延续了很好的SQL使用经验，对于没有那么高的对象模型要求的项目来说，相当完美。

2. 可以进行更为细致的SQL优化，可以减少查询字段。

3. 缺点就是框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改。

4. 二级缓存机制不佳。

hibernate：

1. 功能强大，数据库无关性好，O/R映射能力强，如果你对Hibernate相当精通，而且对Hibernate进行了适当的封装，那么你的项目整个持久层代码会相当简单，需要写的代码很少，开发速度很快，非常爽。

2. 有更好的二级缓存机制，可以使用第三方缓存。

3. 缺点就是学习门槛不低，要精通门槛更高，而且怎么设计O/R映射，在性能和对象模型之间如何权衡取得平衡，以及怎样用好Hibernate方面需要你的经验和能力都很强才行。

举个形象的比喻：

mybatis：机械工具，使用方便，拿来就用，但工作还是要自己来作，不过工具是活的，怎么使由我决定。﻿﻿

hibernate：智能机器人，但研发它（学习、熟练度）的成本很高，工作都可以摆脱他了，但仅限于它能做的事。﻿﻿

通过上篇介绍mybatis与hibernate区别，我们已经能得出一些mybatis的优缺点，但那只是相对于hibernate的，并不全面，我来继续总结mybatis的优缺点，以便大家对于mybatis的了解能更全面些。但我所说的优缺点，仅是我个人总结并结合使用体验后得出的结果，并不能代表大众想法，因此才以“浅谈”作为文章标题。如果大家的见解与我不同，欢迎积极提出来一块讨论，我也借以弥补自己认识的不足和短见。

优点：

1. 易于上手和掌握。

2. sql写在xml里，便于统一管理和优化。

3. 解除sql与程序代码的耦合。

4. 提供映射标签，支持对象与数据库的orm字段关系映射

5. 提供对象关系映射标签，支持对象关系组建维护

6. 提供xml标签，支持编写动态sql。

缺点：

1. sql工作量很大，尤其是字段多、关联表多时，更是如此。

2. sql依赖于数据库，导致数据库移植性差。

3. 由于xml里标签id必须唯一，导致DAO中方法不支持方法重载。

4. 字段映射标签和对象关系映射标签仅仅是对映射关系的描述，具体实现仍然依赖于sql。（比如配置了一对多Collection标签，如果sql里没有join子表或查询子表的话，查询后返回的对象是不具备对象关系的，即Collection的对象为null）

5. DAO层过于简单，对象组装的工作量较大。

6.  不支持级联更新、级联删除。

7. 编写动态sql时,不方便调试，尤其逻辑复杂时。

8 提供的写动态sql的xml标签功能简单（连struts都比不上），编写动态sql仍然受限，且可读性低。

9. 若不查询主键字段，容易造成查询出的对象有“覆盖”现象。

10. 参数的数据类型支持不完善。（如参数为Date类型时，容易报没有get、set方法，需在参数上加@param）

11. 多参数时，使用不方便，功能不够强大。（目前支持的方法有map、对象、注解@param以及默认采用012索引位的方式）

12. 缓存使用不当，容易产生脏数据。

总结：

mybatis的优点其实也是mybatis的缺点，正因为mybatis使用简单，数据的可靠性、完整性的瓶颈便更多依赖于程序员对sql的使用水平上了。sql写在xml里，虽然方便了修改、优化和统一浏览，但可读性很低，调试也非常困难，也非常受限，无法像jdbc那样在代码里根据逻辑实现复杂动态sql拼接。mybatis简单看就是提供了字段映射和对象关系映射的jdbc，省去了数据赋值到对象的步骤而已，除此以外并无太多作为，不要把它想象成hibernate那样强大，简单小巧易用上手，方便浏览修改sql就是它最大的优点了。

mybatis适用于小型且程序员能力较低的项目和人群使用，对于中大型项目来说我并不推荐使用，如果觉得hibernate效率低的话（实际上也是使用不当所致，hibernate是实际上是不适用于拥有高负载的工程项目），还不如直接用spring提供的jdbc简单框架（Template），同样支持对象映射。

# 同一个帐号在一台设备上登录了，在内外一台设备上登录后，前者被挤下线，如何实现？

<https://my.oschina.net/kepler/blog/205674>

# 什么是单点登录?

单点登录Single Sign On简称为SSO，是目前比较流行的企业业务整合的解决方案之一。F5 BIG-IP Edge Gateway解决方案借助SSO改进用户体验。SSO的定义是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统。

企业应用集成（EAI）。企业应用集成可以在不同层面上进行：例如在数据存储层面上的“数据大集中”，在传输层面上的“通用数据交换平台”，在应用层面上的“业务流程整合”，和用户界面上的“通用企业门户”等等。事实上，还用一个层面上的集成变得越来越重要，那就是“身份认证”的整合，也就是“单点登录”。

单点登录的技术实现机制：当用户第一次访问应用系统1的时候，因为还没有登录，会被引导到认证系统中进行登录；

根据用户提供的登录信息，认证系统进行身份效验，如果通过效验，应该返回给用户一个认证的凭据－－ticket；用户再访问别的应用的时候，就会将这个ticket带上，作为自己认证的凭据，应用系统接受到请求之后会把ticket送到认证系统进行效验，检查ticket的合法性。如果通过效验，用户就可以在不用再次登录的情况下访问应用系统2和应用系统3了。

可以看出，要实现SSO，需要以下主要的功能：

所有应用系统共享一个身份认证系统；

所有应用系统能够识别和提取ticket信息；

应用系统能够识别已经登录过的用户，能自动判断当前用户是否登录过，从而完成单点登录的功能。

其中统一的身份认证系统最重要，认证系统的主要功能是将用户的登录信息和用户信息库相比较，对用户进行登录认证；认证成功后，认证系统应该生成统一的认证标志（ticket），返还给用户。另外，认证系统还应该对ticket进行效验，判断其有效性。整个系统可以存在两个以上的认证服务器，这些服务器甚至可以是不同的产品。认证服务器之间要通过标准的通讯协议，互相交换认证信息，就能完成更高级别的单点登录。

F5 BIG-IP Edge Gateway解决方案借助SSO改进用户体验，简化移动/远程员工的身份验证流程不仅能够提升用户体验，还能够最大限度降低安全风险，并减少密码锁闭呼叫帮助中心的次数。F5 BIG-IP Edge Gateway能够缓存登录证书，并支持在登录过程中身份验证。如果发生连接掉线，用户将被自动重新执行身份验证。

# 什么是AOP?

 AOP: (Aspect Oriented Programming) 面向切面编程。是目前软件开发中的一个热点，也是**[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/hz_blog/article/details/_blank)**框架中容。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。

             AOP、OOP在字面上虽然非常类似，但却是面向不同领域的两种设计思想。OOP（面向对象编程）针对业务处理过程的实体及其属性和行为进行抽象封装，以获得更加清晰高效的逻辑单元划分。 而AOP则是针对业务处理过程中的切面进行提取，它所面对的是处理过程中的某个步骤或阶段，以获得逻辑过程中各部分之间低耦合性的隔离效果。这两种设计思想在目标上有着本质的差异。

　　     举个简单的例子，对于“雇员”这样一个业务实体进行封装，自然是OOP的任务，我们可以为其建立一个“Employee”类，并将“雇员”相关的属性和行为封装其中。而用AOP设计思想对“雇员”进行封装将无从谈起。

　　    同样，对于“权限检查”这一动作片断进行划分，则是AOP的目标领域。而通过OOP对一个动作进行封装，则有点不伦不类。 换而言之，OOP面向名词领域，AOP面向动词领域。

           总之，AOP可以通过预编译方式和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。

# 接口和抽象类的区别?

抽象类和接口都不能直接实例化，如果要实例化，抽象类变量必须指向实现所有抽象方法的子类对象，接口变量必须指向实现所有接口方法的类对象。

2、抽象类要被子类继承，接口要被类实现。

3、接口只能做方法申明，抽象类中可以做方法申明，也可以做方法实现

4、接口里定义的变量只能是公共的静态的常量，抽象类中的变量是普通变量。

5、抽象类里的抽象方法必须全部被子类所实现，如果子类不能全部实现父类抽象方法，那么该子类只能是抽象类。同样，一个实现接口的时候，如不能全部实现接口方法，那么该类也只能为抽象类。

6、抽象方法只能申明，不能实现，接口是设计的结果 ，抽象类是重构的结果

7、抽象类里可以没有抽象方法

8、如果一个类里有抽象方法，那么这个类只能是抽象类

9、抽象方法要被实现，所以不能是静态的，也不能是私有的。

10、接口可继承接口，并可多继承接口，但类只能单根继承。

1.抽象类 和 接口 都是用来抽象具体对象的. 但是接口的抽象级别最高

2.抽象类可以有具体的方法 和属性, 接口只能有抽象方法和不可变常量

3.抽象类主要用来抽象类别,接口主要用来抽象功能.

4、抽象类中，且不包含任何实现，派生类必须覆盖它们。接口中所有方法都必须是未实现的。

# 数据精度丢失问题?

# Sql优化方案

# **登录注册用到哪些校验方式?**

# **对于java中的反射机制的理解?**

# **对IOC 和 AOP 的理解**

# SpringMVC有几种bean?

1 . 一类是使用Bean，即是把已经在xml文件中配置好的Bean拿来用，完成属性、方法的组装；比如@Autowired , @Resource，可以通过byTYPE（@Autowired）、byNAME（@Resource）的方式获取Bean；

2、一类是注册Bean,@Component , @Repository , @ Controller , @Service , @Configration这些注解都是把你要实例化的对象转化成一个Bean，放在IoC容器中，等你要用的时候，它会和上面的@Autowired , @Resource配合到一起，把对象、属性、方法完美组装。

# Jquery有哪些选择器?

# 如何解决多线程并发问题?

# 你在数据库操作是最多关联几张表?

# 在mysql分页查询时用的limit(参数1,参数2)中参数1是? , 参数2是?

limit是mysql的语法  
select \* from table limit m,n  
其中m是指记录开始的index，[从0开始](https://www.baidu.com/s?wd=%E4%BB%8E0%E5%BC%80%E5%A7%8B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1mvfkP10snWPhuHD3P10k0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWmvPjckrjn1" \t "https://zhidao.baidu.com/question/_blank)，表示第一条记录  
n是指从第m+1条开始，取n条。  
select \* from tablename limit 2,4  
即取出第3条至第6条，4条记录

# 对分布式框架的理解?

# 购物平台的所用到的一些技术和开发思路

# java开发的23种开发模式中你对哪几种模式比较熟悉?说说其原理．

# 购物车开发中如何改变一件商品的状态？

# Mybatis的工作原理？

## 什么是Mybatis

       MyBatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。iBATIS一词来源于“internet”和“abatis”的组合，是一个基于**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)**的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）。

      MyBatis参考资料官网：[https://mybatis.github.io/mybatis-3/zh/index.html](https://mybatis.github.io/mybatis-3/zh/index.html" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)

官网对Mybatis的介绍更加具有权威性：

**[java]** [view plain](http://blog.csdn.net/jiuqiyuliang/article/details/45286191" \o "view plain" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank) [copy](http://blog.csdn.net/jiuqiyuliang/article/details/45286191" \o "copy" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank) [print](http://blog.csdn.net/jiuqiyuliang/article/details/45286191" \o "print" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)[?](http://blog.csdn.net/jiuqiyuliang/article/details/45286191" \o "?" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)

1. MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手工设置参数以及抽取结果集。MyBatis 使用简单的 XML 或注解来配置和映射基本体，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

       MyBatis是iBatis的升级版，用法有很多的相似之处，但是MyBatis进行了重要的改进。例如：

**1、Mybatis实现了接口绑定，使用更加方便。**

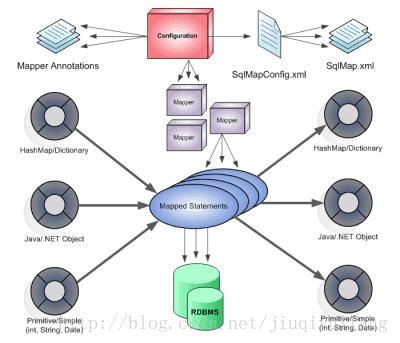
      在ibatis2.x中我们需要在DAO的实现类中指定具体对应哪个xml映射文件， 而Mybatis实现了DAO接口与xml映射文件的绑定，自动为我们生成接口的具体实现，使用起来变得更加省事和方便。

**2、对象关系映射的改进，效率更高**

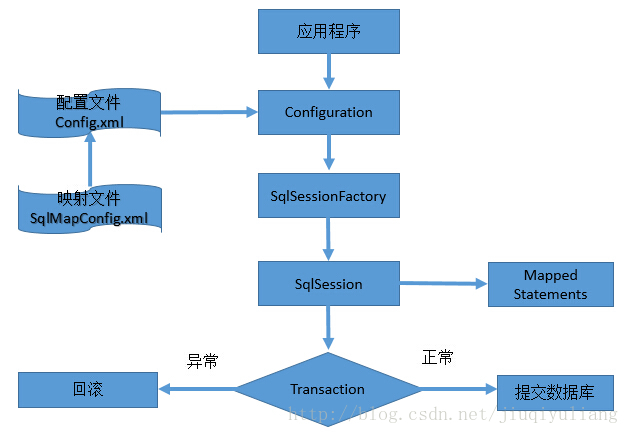
**3、MyBatis采用功能强大的基于OGNL的表达式来消除其他元素。**

       对于IBatis与MyBatis的区别就不再多说了，感兴趣的朋友可以研究一下。

## MyBatis的框架架构



        看到Mybatis的框架图，可以清晰的看到Mybatis的整体核心对象，我更喜欢用自己的图来表达Mybatis的整个的执行流程。如下图所示：



**原理详解：**

        MyBatis应用程序根据XML配置文件创建SqlSessionFactory，SqlSessionFactory在根据配置，配置来源于两个地方，一处是配置文件，一处是Java代码的注解，获取一个SqlSession。SqlSession包含了执行sql所需要的所有方法，可以通过SqlSession实例直接运行映射的sql语句，完成对数据的增删改查和事务提交等，用完之后关闭SqlSession。

## MyBatis的优缺点

### 优点：

1、简单易学

       mybatis本身就很小且简单。没有任何第三方依赖，最简单安装只要两个jar文件+配置几个sql映射文件易于学习，易于使用，通过文档和源代码，可以比较完全的掌握它的设计思路和实现。

2、灵活

       mybatis不会对应用程序或者**[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)**的现有设计强加任何影响。 sql写在xml里，便于统一管理和优化。通过sql基本上可以实现我们不使用数据访问框架可以实现的所有功能，或许更多。

3、解除sql与程序代码的耦合

       通过提供DAL层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)**。sql和代码的分离，提高了可维护性。

4、提供映射标签，支持对象与数据库的orm字段关系映射

5、提供对象关系映射标签，支持对象关系组建维护

6、提供xml标签，支持编写动态sql。

### 缺点：

1、编写SQL语句时工作量很大，尤其是字段多、关联表多时，更是如此。

2、SQL语句依赖于数据库，导致数据库移植性差，不能更换数据库。

3、框架还是比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是要自己写的，工作量也比较大，而且不太容易适应快速数据库修改。

4、二级缓存机制不佳

## 总结

       mybatis的优点同样是mybatis的缺点，正因为mybatis使用简单，数据的可靠性、完整性的瓶颈便更多依赖于程序员对sql的使用水平上了。sql写在xml里，虽然方便了修改、优化和统一浏览，但可读性很低，调试也非常困难，也非常受限。

      mybatis没有**[hibernate](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/jiuqiyuliang/article/details/_blank)**那么强大，但是mybatis最大的优点就是简单小巧易于上手，方便浏览修改sql语句。

# 请描述数据库连接池的工作机制？

服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

# 请描述ＡrrayList，Vector, LinkedList的储存性能和特性?

ArrayList 采用的是数组形式来保存对象的，这种方式将对象放在连续的位置中，所以最大的缺点就是插入删除时非常麻烦

LinkedList 采用的将对象存放在独立的空间中，而且在每个空间中还保存下一个链接的索引 但是缺点就是查找非常麻烦 要丛第一个索引开始

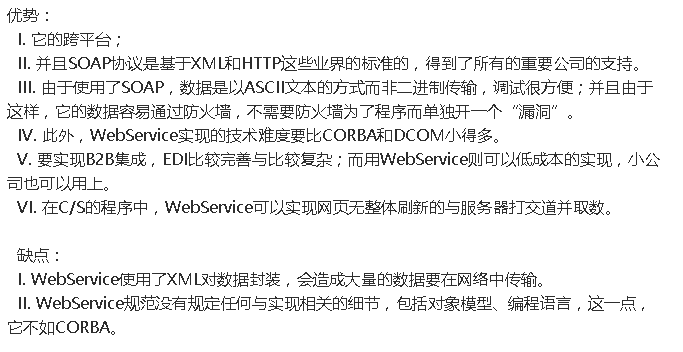
ArrayList和Vector都是用数组方式存储数据,此数组元素数要大于实际的存储空间以便进行元素增加和插入操作,他们都允许直接用序号索引元素,但是插入数据元素涉及到元素移动等内存操作,所以索引数据快而插入数据慢.

Vector使用了sychronized方法(线程安全),所以在性能上比ArrayList要差些.

LinkedList使用双向链表方式存储数据,按序号索引数据需要前向或后向遍历数据，所以索引数据慢,是插入数据时只需要记录前后项即可,所以插入的速度快.

**arraylist和vector的区别?**   
 1).同步性:Vector是线程安全的，也就是说是同步的，而ArrayList是线程不安全的，不是同步的   
 2).数据增长:当需要增长时,Vector默认增长为原来一培，而ArrayList却是原来的一半

# 请描述Web Service 的原理,以及优势和缺点?



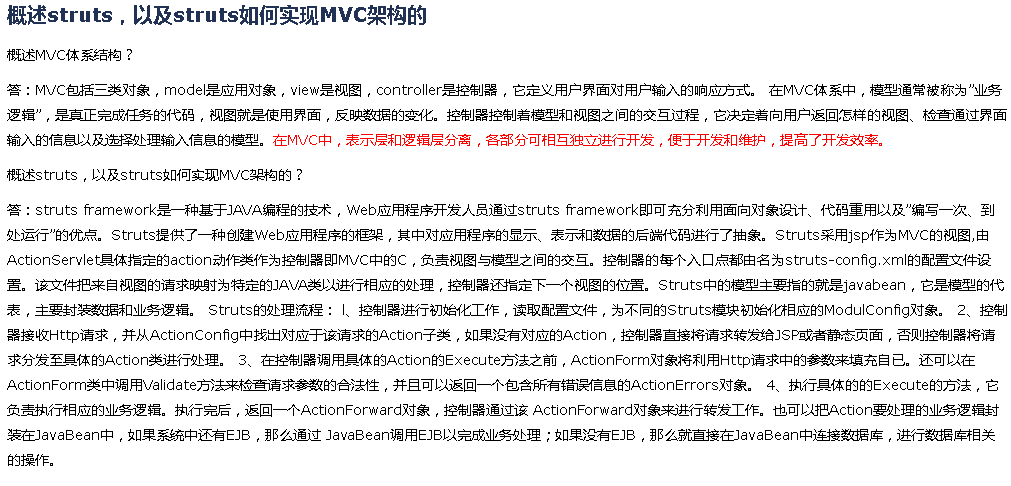
# 请描述Hashtable的原理,以及HashMap和Hashtable的区别?

Hashtable 简介  
和HashMap一样，Hashtable 也是一个散列表，它存储的内容是键值对(key-value)映射。  
Hashtable 继承于Dictionary，实现了Map、Cloneable、IMG_256Java.io.Serializable接口。  
Hashtable 的函数都是同步的，这意味着它是线程安全的。它的key、value都不可以为null。此外，Hashtable中的映射不是有序的。  
  
Hashtable 的实例有两个参数影响其性能：初始容量 和 加载因子。容量 是哈希表中桶 的数量，初始容量 就是哈希表创建时的容量。注意，哈希表的状态为 open：在发生“哈希冲突”的情况下，单个桶会存储多个条目，这些条目必须按顺序搜索。加载因子 是对哈希表在其容量自动增加之前可以达到多满的一个尺度。初始容量和加载因子这两个参数只是对该实现的提示。关于何时以及是否调用 rehash 方法的具体细节则依赖于该实现。  
通常，默认加载因子是 0.75, 这是在时间和空间成本上寻求一种折衷。加载因子过高虽然减少了空间开销，但同时也增加了查找某个条目的时间（在大多数 Hashtable 操作中，包括 get 和 put 操作，都反映了这一点）。  
Hashtable和HashMap类有三个重要的不同之处。第一个不同主要是历史原因。Hashtable是基于陈旧的Dictionary类的，HashMap是Java 1.2引进的Map接口的一个实现。  
　　也许最重要的不同是Hashtable的方法是同步的，而HashMap的方法不是。这就意味着，虽然你可以不用采取任何特殊的行为就可以在一个多线程的应用程序中用一个Hashtable，但你必须同样地为一个HashMap提供外同步。一个方便的方法就是利用Collections类的静态的synchronizedMap()方法，它创建一个线程安全的Map对象，并把它作为一个封装的对象来返回。这个对象的方法可以让你同步访问潜在的HashMap。这么做的结果就是当你不需要同步时，你不能切断Hashtable中的同步（比如在一个单线程的应用程序中），而且同步增加了很多处理费用。  
　　第三点不同是，只有HashMap可以让你将空值作为一个表的条目的key或value。HashMap中只有一条记录可以是一个空的key，但任意数量的条目可以是空的value。这就是说，如果在表中没有发现搜索键，或者如果发现了搜索键，但它是一个空的值，那么get()将返回null。如果有必要，用containKey()方法来区别这两种情况。

# 请概述MVC体系结构?

MVC包括三类对象，model是应用对象，view是视图，controller是控制器，它定义用户界面对用户输入的响应方式。在MVC体系中，模型通常被称为“业务逻辑”，是真正完成任务的代码，视图就是使用界面，反映数据的变化。控制器控制着模型和视图之间的交互过程，它决定着向用户返回怎样的视图、检查通过界面输入的信息以及选择处理输入信息的模型.在MVC中，表示层和逻辑层分离，各部分可相互独立进行开发，便于开发和维护，提高了开发效率。

# 请概述struts,以及struts如何实现MVC架构的?



# 请概述java中的回收机制?

在java编程中，垃圾回收机制是JVM默认执行的，不是我们程序员所能控制的，这点比C要高级，在C中，垃圾回收完全是由程序员控制的。

java中我们可以调用System.gc();方法来让JVM(java虚拟机执行垃圾回收)，但是什么时候执行还是靠JAVA的JVM来自动执行的。

我们可以让变量、常量、用完后指向null，这样就是一个null的垃圾，JVM执行的时候就会回收。也就是说我们可以利用指向null来通知JVM这个事没用的变量！

# oracle的函数和存储过程有什么区别

# Spring的事物控制

# Map、List、Set的区别?

List,Set,Map是否继承自Collection接口？   
答：List，Set是，Map不是。 Collection是最基本的集合接口，一个Collection代表一组Object，即Collection的元素。一些Collection允许相同的元素而另一些不行。一些能排序而另一些不行。Java JDK不能提供直接继承自Collection的类，Java JDK提供的类都是继承自Collection的"子接口"，如:List和Set。   
注意：Map没有继承Collection接口，Map提供key到value的映射。一个Map中不能包含相同key，每个key只能映射一个value。Map接口提供3种集合的视图，Map的内容可以被当做一组key集合，一组value集合，或者一组key-value映射。

List按对象进入的顺序保存对象，不做排序或编辑操作。Set对每个对象只接受一次，并使用自己内部的排序方法(通常，你只关心某个元素是否属于Set,而不关心它的顺序--否则应该使用List)。Map同样对每个元素保存一份，但这是基于"键"的，Map也有内置的排序，因而不关心元素添加的顺序。如果添加元素的顺序对你很重要，应该使用 LinkedHashSet或者LinkedHashMap.

详细介绍：   
List特点：元素有放入顺序，元素可重复   
Map特点：元素按键值对存储，无放入顺序   
Set特点：元素无放入顺序，元素不可重复（注意：元素虽然无放入顺序，但是元素在set中的位置是有该元素的HashCode决定的，其位置其实是固定的）   
List接口有三个实现类：LinkedList，ArrayList，Vector   
LinkedList：底层基于链表实现，链表内存是散乱的，每一个元素存储本身内存地址的同时还存储下一个元素的地址。链表增删快，查找慢   
ArrayList和Vector的区别：ArrayList是非线程安全的，效率高；Vector是基于线程安全的，效率低   
Set接口有两个实现类：HashSet(底层由HashMap实现)，LinkedHashSet   
SortedSet接口有一个实现类：TreeSet（底层由平衡二叉树实现）   
Query接口有一个实现类：LinkList   
Map接口有三个实现类：HashMap，HashTable，LinkeHashMap   
  HashMap非线程安全，高效，支持null；HashTable线程安全，低效，不支持null   
SortedMap有一个实现类：TreeMap   
其实最主要的是，list是用来处理序列的，而set是用来处理集的。Map是知道的，存储的是键值对   
set 一般无序不重复.map kv 结构 list 有序 。

**List的功能方法**

　　实际上有两种List: 一种是基本的ArrayList,其优点在于随机访问元素，另一种是更强大的LinkedList,它并不是为快速随机访问设计的，而是具有一套更通用的方法。

　　List : 次序是List最重要的特点：它保证维护元素特定的顺序。List为Collection添加了许多方法，使得能够向List中间插入与移除元素(这只推荐LinkedList使用。)一个List可以生成ListIterator,使用它可以从两个方向遍历List,也可以从List中间插入和移除元素。

　　ArrayList : 由数组实现的List。允许对元素进行快速随机访问，但是向List中间插入与移除元素的速度很慢。ListIterator只应该用来由后向前遍历ArrayList,而不是用来插入和移除元素。因为那比LinkedList开销要大很多。

　　LinkedList : 对顺序访问进行了优化，向List中间插入与删除的开销并不大。随机访问则相对较慢。(使用ArrayList代替。)还具有下列方法：addFirst(), addLast(), getFirst(), getLast(), removeFirst() 和 removeLast(), 这些方法 (没有在任何接口或基类中定义过)使得LinkedList可以当作堆栈、队列和双向队列使用。

**Set的功能方法**

　　Set具有与Collection完全一样的接口，因此没有任何额外的功能，不像前面有两个不同的List。实际上Set就是Collection,只是行为不同。(这是继承与多态思想的典型应用：表现不同的行为。)Set不保存重复的元素(至于如何判断元素相同则较为负责)

　　Set : 存入Set的每个元素都必须是唯一的，因为Set不保存重复元素。加入Set的元素必须定义equals()方法以确保对象的唯一性。Set与Collection有完全一样的接口。Set接口不保证维护元素的次序。

　　HashSet : 为快速查找设计的Set。存入HashSet的对象必须定义hashCode()。

　　TreeSet : 保存次序的Set, 底层为树结构。使用它可以从Set中提取有序的序列。

　　LinkedHashSet : 具有HashSet的查询速度，且内部使用链表维护元素的顺序(插入的次序)。于是在使用迭代器遍历Set时，结果会按元素插入的次序显示。

**Map的功能方法**

　　方法put(Object key, Object value)添加一个“值”(想要得东西)和与“值”相关联的“键”(key)(使用它来查找)。方法get(Object key)返回与给定“键”相关联的“值”。可以用containsKey()和containsValue()**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "http://blog.csdn.net/vstar283551454/article/details/_blank)**Map中是否包含某个“键”或“值”。标准的**[Java](http://lib.csdn.net/base/javase" \o "Java SE知识库" \t "http://blog.csdn.net/vstar283551454/article/details/_blank)**类库中包含了几种不同的Map：HashMap, TreeMap, LinkedHashMap, WeakHashMap, IdentityHashMap。它们都有同样的基本接口Map，但是行为、效率、排序策略、保存对象的生命周期和判定“键”等价的策略等各不相同。

　　执行效率是Map的一个大问题。看看get()要做哪些事，就会明白为什么在ArrayList中搜索“键”是相当慢的。而这正是HashMap提高速度的地方。HashMap使用了特殊的值，称为“散列码”(hash code)，来取代对键的缓慢搜索。“散列码”是“相对唯一”用以代表对象的int值，它是通过将该对象的某些信息进行转换而生成的。所有Java对象都能产生散列码，因为hashCode()是定义在基类Object中的方法。

　　HashMap就是使用对象的hashCode()进行快速查询的。此方法能够显著提高性能。

　　Map : 维护“键值对”的关联性，使你可以通过“键”查找“值”

　　HashMap : Map基于散列表的实现。插入和查询“键值对”的开销是固定的。可以通过构造器设置容量capacity和负载因子load factor，以调整容器的性能。

　　LinkedHashMap : 类似于HashMap，但是迭代遍历它时，取得“键值对”的顺序是其插入次序，或者是最近最少使用(LRU)的次序。只比HashMap慢一点。而在迭代访问时发而更快，因为它使用链表维护内部次序。

　　TreeMap : 基于红黑树**[数据结构](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/vstar283551454/article/details/_blank)**的实现。查看“键”或“键值对”时，它们会被排序(次序由Comparabel或Comparator决定)。TreeMap的特点在于，你得到的结果是经过排序的。TreeMap是唯一的带有subMap()方法的Map，它可以返回一个子树。

　　WeakHashMao : 弱键(weak key)Map，Map中使用的对象也被允许释放: 这是为解决特殊问题设计的。如果没有map之外的引用指向某个“键”，则此“键”可以被垃圾收集器回收。

　　IdentifyHashMap : 使用==代替equals()对“键”作比较的hash map。专为解决特殊问题而设计。

# 如何将一个Map/List放入另外一个Map/List ?

# SpringMVC 的流程

# Spring的一些常用注解

# SVN的作用及一些功能操作,举例说明.

# 重写和重载的区别?

1)重写:(override)

        a)父子类中,方法名相同,参数列表相同

        b)遵循"运行期"绑定,根据对象的类型调用方法

2)重载:(overload):

        a)同一个(或父子)类中,方法名相同,参数列表不同

        b)遵循"编译期"绑定,根据引用的类型绑定方法  
两者都是多态的表现  
Overload(重载)，方法名相同，但参数类型，个数，顺序不同为重载。（是同一类中方法之间的区别），

Override(重写)，子类重写父类，子类的方法名与父类名一致，参数个数，类型都与父类完全一样，为方法的重写。（是类与类，父子类中之间的区别）  
重写遵循一同两小一大原则:

两同：1)方法名；2)形参列表  
两小：1)返回值类型比父类更小或相等； 2)异常比父类方法更小或相等  
一大：子类权限比父类大或相等