

# 如何实现一个简单的RPC



柳树之 (/u/86696f09d988)

2018.05.09 08:05 字数 2756 阅读 37437 评论 26 喜欢 335

(/u/86696f09d988)

在如何给老婆解释什么是RPC (<https://www.jianshu.com/p/2accc2840a1b>)中，我们讨论了RPC的实现思路。

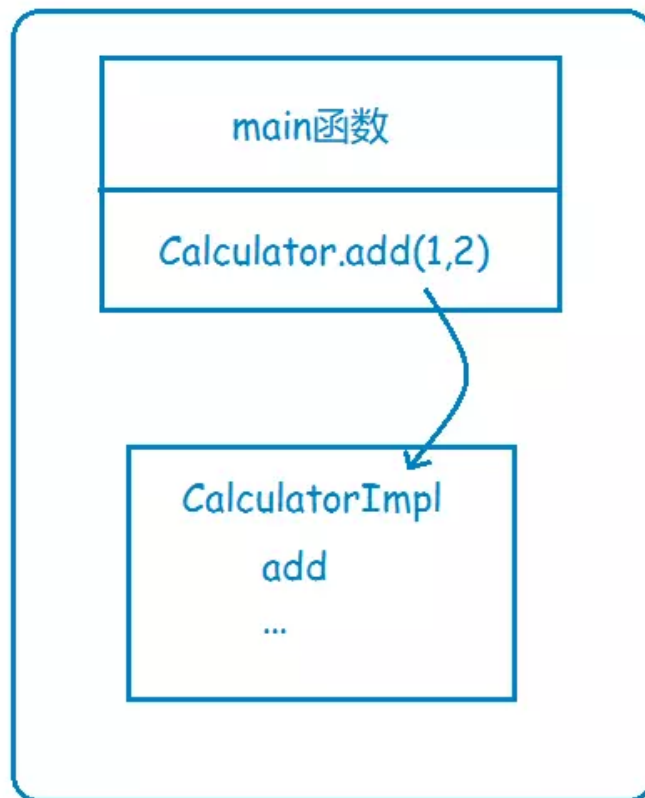
那么这一次，就让我们通过代码来实现一个简单的RPC吧！

## RPC的实现原理

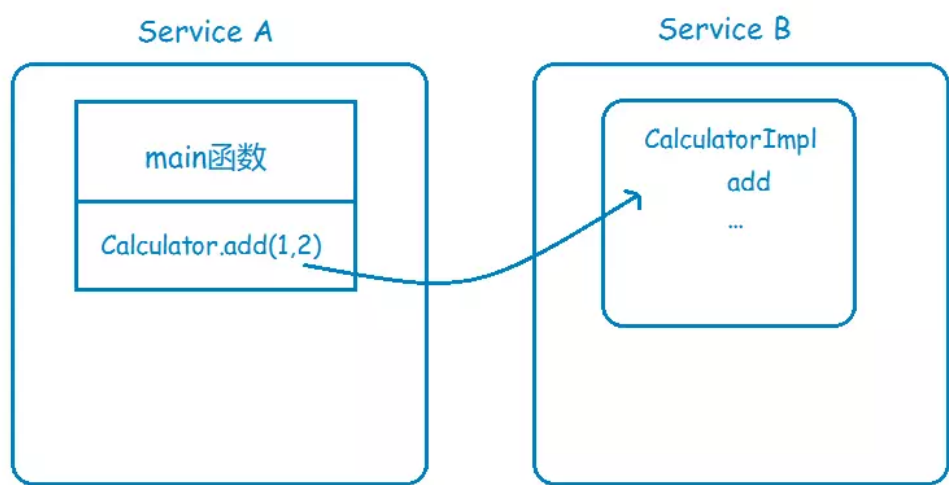
正如上一讲所说，RPC主要是为了解决的两个问题：

- 解决分布式系统中，服务之间的调用问题。
- 远程调用时，要能够像本地调用一样方便，让调用者感知不到远程调用的逻辑。

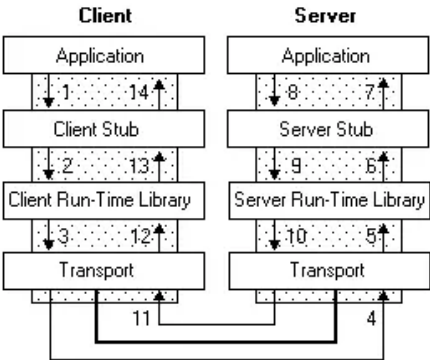
还是以计算器Calculator为例，如果实现类CalculatorImpl是放在本地的，那么直接调用即可：



现在系统变成分布式了，CalculatorImpl和调用方不在同一个地址空间，那么就必须要进行远程过程调用：



那么如何实现远程过程调用，也就是RPC呢，一个完整的RPC流程，可以用下面这张图来描述：



其中左边的Client，对应的就是前面的Service A，而右边的Server，对应的则是Service B。

下面一步一步详细解释一下。

1. Service A的应用层代码中，调用了Calculator的一个实现类的add方法，希望执行一个加法运算；
2. 这个Calculator实现类，内部并不是直接实现计算器的加减乘除逻辑，而是通过远程调用Service B的RPC接口，来获取运算结果，因此称之为**Stub**；
3. Stub怎么和Service B建立远程通讯呢？这时候就要用到**远程通讯工具**了，也就是图中的**Run-time Library**，这个工具将帮你实现远程通讯的功能，比如Java的**Socket**，就是这样一个库，当然，你也可以用基于Http协议的**HttpClient**，或者其他通讯工具类，都可以，**RPC并没有规定说你要用何种协议进行通讯**；
4. Stub通过调用通讯工具提供的方法，和服务 B 建立起了通讯，然后将请求数据发给 Service B。需要注意的是，由于底层的网络通讯是基于**二进制格式**的，因此这里Stub

+

🔖

❤

🔗

- 传给通讯工具类的数据也必须是二进制，比如`calculator.add(1,2)`，你必须把参数值1和2放到一个Request对象里头（这个Request对象当然不只这些信息，还包括要调用哪个服务的哪个RPC接口等其他信息），然后**序列化为二进制**，再传给通讯工具类，这一点也将在下面的代码实现中体现；
5. 二进制的数据传到Service B这一边了，Service B当然也有自己的通讯工具，通过这个通讯工具接收二进制的请求；
  6. 既然数据是二进制的，那么自然要进行**反序列化**了，将二进制的数据反序列化为请求对象，然后将这个请求对象交给Service B的Stub处理；
  7. 和之前的Service A的Stub一样，这里的Stub也同样是个“假玩意”，它所负责的，只是去解析请求对象，知道调用方要调的是哪个RPC接口，传进来的参数又是什么，然后再把这些参数传给对应的RPC接口，也就是Calculator的实际实现类去执行。很明显，如果是Java，那这里肯定用到了**反射**。
  8. RPC接口执行完毕，返回执行结果，现在轮到Service B要把数据发给Service A了，怎么发？一样的道理，一样的流程，只是现在Service B变成了Client，Service A变成了Server而已：Service B反序列化执行结果->传输给Service A->Service A反序列化执行结果->将结果返回给Application，完毕。

理论的讲完了，是时候把理论变成实践了。

## 把理论变成实践

本文的示例代码，可到Github (<https://link.jianshu.com?t=https%3A%2F%2Fgithub.com%2Fhzy38324%2Fsimple-rpc>)下载。

首先是Client端的应用层怎么发起RPC，ConsumerApp：

```
public class ConsumerApp {
    public static void main(String[] args) {
        Calculator calculator = new CalculatorRemoteImpl();
        int result = calculator.add(1, 2);
    }
}
```

**通过一个CalculatorRemoteImpl，我们把RPC的逻辑封装进去了，客户端调用时感知不到远程调用的麻烦。**下面再来看看CalculatorRemoteImpl，代码有些多，但是其实就是把上面的2、3、4几个步骤用代码实现了而已，CalculatorRemoteImpl：



```
public class CalculatorRemoteImpl implements Calculator {
    public int add(int a, int b) {
        List<String> addressList = lookupProviders("Calculator.add");
        String address = chooseTarget(addressList);
        try {
            Socket socket = new Socket(address, PORT);

            // 将请求序列化
            CalculateRpcRequest calculateRpcRequest = generateRequest(a, b);
            ObjectOutputStream objectOutputStream = new ObjectOutputStream(socket.getOutputStream());

            // 将请求发给服务提供方
            objectOutputStream.writeObject(calculateRpcRequest);

            // 将响应体反序列化
            ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream());
            Object response = objectInputStream.readObject();

            if (response instanceof Integer) {
                return (Integer) response;
            } else {
                throw new InternalError();
            }
        } catch (Exception e) {
            log.error("fail", e);
            throw new InternalError();
        }
    }
}
```

add方法的前面两行，lookupProviders和chooseTarget，可能大家会觉得不明觉厉。

分布式应用下，一个服务可能有多个实例，比如Service B，可能有ip地址为198.168.1.11和198.168.1.13两个实例，lookupProviders，其实就是在寻找要调用的服务的实例列表。在分布式应用下，通常会有一个**服务注册中心**，来提供查询实例列表的功能。

查到实例列表之后要调用哪一个实例呢，只时候就需要chooseTarget了，其实内部就是一个**负载均衡策略**。

由于我们这里只是想实现一个简单的RPC，所以暂时不考虑服务注册中心和负载均衡，因此代码里写死了返回ip地址为127.0.0.1。

代码继续往下走，我们这里用到了Socket来进行远程通讯，同时利用**ObjectOutputStream**的writeObject和**ObjectInputStream**的readObject，来实现序列化和反序列化。

最后再来看看Server端的实现，和Client端非常类似，ProviderApp：



```

public class ProviderApp {
    private Calculator calculator = new CalculatorImpl();

    public static void main(String[] args) throws IOException {
        new ProviderApp().run();
    }

    private void run() throws IOException {
        ServerSocket listener = new ServerSocket(9090);
        try {
            while (true) {
                Socket socket = listener.accept();
                try {
                    // 将请求反序列化
                    ObjectInputStream objectInputStream = new ObjectInputStream(socket.getInputStream());
                    Object object = objectInputStream.readObject();

                    log.info("request is {}", object);

                    // 调用服务
                    int result = 0;
                    if (object instanceof CalculateRpcRequest) {
                        CalculateRpcRequest calculateRpcRequest = (CalculateRpcRequest) object;
                        if ("add".equals(calculateRpcRequest.getMethod())) {
                            result = calculator.add(calculateRpcRequest.getA(), calculateRpcRequest.getB());
                        } else {
                            throw new UnsupportedOperationException();
                        }
                    }

                    // 返回结果
                    ObjectOutputStream objectOutputStream = new ObjectOutputStream(socket.getOutputStream());
                    objectOutputStream.writeObject(new Integer(result));
                } catch (Exception e) {
                    log.error("fail", e);
                } finally {
                    socket.close();
                }
            }
        } finally {
            listener.close();
        }
    }
}

```

Server端主要是通过ServerSocket的accept方法，来接收Client端的请求，接着就是反序列化请求->执行->序列化执行结果，最后将二进制格式的执行结果返回给Client。

**就这样我们实现了一个简陋而又详细的RPC。**

说它简陋，是因为这个实现确实比较挫，在下一小节会说它为什么挫。

说它详细，是因为它一步一步的演示了一个RPC的执行流程，方便大家了解RPC的内部机制。

## 为什么说这个RPC实现很挫

这个RPC实现只是为了给大家演示一下RPC的原理，要是想放到生产环境去用，那是绝对不行的。



### 1、缺乏通用性

我通过给Calculator接口写了一个CalculatorRemoteImpl，来实现计算器的远程调用，下一次要是有别的接口需要远程调用，是不是又得再写对应的远程调用实现类？这肯定是很不方便的。

那该如何解决呢？先来看看使用Dubbo时是如何实现RPC调用的：

```
@Reference
private Calculator calculator;

...

calculator.add(1,2);

...
```

Dubbo通过和Spring的集成，在Spring容器初始化的时候，如果扫描到对象加了@Reference注解，那么就给这个对象生成一个代理对象，这个代理对象会负责远程通讯，然后将代理对象放进容器中。所以代码运行期用到的calculator就是那个代理对象了。

我们可以先不和Spring集成，也就是先不采用依赖注入，但是我们要做到像Dubbo一样，无需自己手动写代理对象，怎么做呢？那自然是要求所有的远程调用都遵循一套模板，把远程调用的信息放到一个RpcRequest对象里面，发给Server端，Server端解析之后就知道你要调用的是哪个RPC接口、以及入参是什么类型、入参的值又是什么，就像Dubbo的RpcInvocation：

```
public class RpcInvocation implements Invocation, Serializable {

    private static final long serialVersionUID = -4355285085441097045L;

    private String methodName;

    private Class<?>[] parameterTypes;

    private Object[] arguments;

    private Map<String, String> attachments;

    private transient Invoker<?> invoker;
```

### 2、集成Spring

在实现了代理对象通用化之后，下一步就可以考虑集成Spring的IOC功能了，通过Spring来创建代理对象，这一点就需要对Spring的bean初始化有一定掌握了。

### 3、长连接or短连接

总不能每次要调用RPC接口时都去开启一个Socket建立连接吧？是不是可以保持若干个长连接，然后每次有rpc请求时，把请求放到任务队列中，然后由线程池去消费执行？只是一个思路，后续可以参考一下Dubbo是如何实现的。



#### 4、服务端线程池

我们现在的Server端，是单线程的，每次都要等一个请求处理完，才能去accept另一个socket的连接，这样性能肯定很差，是不是可以通过一个线程池，来实现同时处理多个RPC请求？同样只是一个思路。

#### 5、服务注册中心

正如之前提到的，要调用服务，首先你需要一个服务注册中心，告诉你对方服务都有哪些实例。Dubbo的服务注册中心是可以配置的，官方推荐使用Zookeeper。如果使用Zookeeper的话，要怎样往上面注册实例，又要怎样获取实例，这些都是要实现的。

#### 6、负载均衡

如何从多个实例里挑选一个出来，进行调用，这就要用到负载均衡了。负载均衡的策略肯定不只一种，要怎样把策略做成可配置的？又要如何实现这些策略？同样可以参考Dubbo，Dubbo - 负载均衡 (<https://link.jianshu.com?t=http%3A%2F%2Fdubbo.apache.org%2Fbooks%2Fdubbo-user-book%2Fdemos%2Floadbalance.html>)

#### 7、结果缓存

每次调用查询接口时都要真的去Server端查询吗？是不是要考虑一下支持缓存？

#### 8、多版本控制

服务端接口修改了，旧的接口怎么办？

#### 9、异步调用

客户端调用完接口之后，不想等待服务端返回，想去干点别的事，可以支持不？

#### 10、优雅停机

服务端要停机了，还没处理完的请求，怎么办？

.....

诸如此类的优化点还有很多，这也是为什么实现一个高性能高可用的RPC框架那么难的原因。

当然，我们现在已经有很多很不错的RPC框架可以参考了，我们完全可以借鉴一下前人的智慧。

**后面如果有 ( dian ) 机 ( zan ) 会 ( duo ) 的话，也将和大家分享一下如何一步一步优化现有的这块RPC代码，把它做成一个小型RPC框架！**

## 参考

- 一本很棒的分布式书籍：《大型网站系统与Java中间件实践》
- Dubbo 使用文档 (<https://link.jianshu.com?t=http%3A%2F%2Fdubbo.apache.org%2Fbooks%2Fdubbo-user-book%2F>)



- Dubbo 源码开发手册 (<https://link.jianshu.com?t=http%3A%2F%2Fdubbo.apache.org%2Fbooks%2Fdubbo-dev-book%2F>)

点赞就是最好的赞赏

赞赏支持

📖 技术之路 (/nb/16156802) 举报文章 © 著作权归作者所有



柳树之 (/u/86696f09d988) 

写了 150374 字，被 18904 人关注，获得了 4210 个喜欢 (/u/86696f09d988)

+ 关注

用简单的语言，分享心得和感受。 个人公众号 => Bridge4You 个人博客 => <http://bridgeforyou.cn> 知乎 => h...

喜欢 336



更多分享

开发10年  
全记在这本Java进阶宝典了

Spring源码分析

分布式架构

微服务架构

JVM性能优化


高效DevOps

多线程并发编程

点击领取





(/p/428251ede1aa)



写下你的评论...


精彩评论 ( 2 )



柳树之 (/u/86696f09d988) 


2楼 · 2018.05.09 11:27 (/u/86696f09d988)

寄己评论寄己，哼！


9人赞  回复

苦海行舟 (/u/68bd548592e8) : 🤔 哈哈





2018.05.09 15:20  回复

洋河蓝 (/u/5327eee9d28c) : @SexyCode (/users/86696f09d988) 哪里人啊，茂名吗

2018.05.16 22:33  回复

何時 (/u/c3894681fe43) : 通俗易懂，大佬，给你120个赞！！👍

2018.08.21 10:55  回复

 添加新评论



逃离魔爪 (/u/3a73297e96ad)

5楼 · 2018.05.22 17:28

(/u/3a73297e96ad)

大佬写的真是通俗易懂，有的人技术牛，但是表达又不是很好，你是属于两者兼备的，希望能出后续的文章，把RPC框架完善啊

4人赞  回复

26条评论

只看作者

按时间倒序 按时间正序



蒲公英的梦想\_e360 (/u/c77522cba82c)

23楼 · 2018.12.28 09:33

(/u/c77522cba82c)

好文章，通俗易懂

赞  回复



小酒窝\_5be9 (/u/b4468e972422)

22楼 · 2018.12.27 21:33

(/u/b4468e972422)

真棒，易懂，非常感谢

赞  回复



丢弃幸福猪 (/u/a80d71d02950)

21楼 · 2018.11.21 18:10

(/u/a80d71d02950)

逻辑清楚，简单易懂，谢谢

赞  回复



大数据工程师 (/u/5de885cec043)

20楼 · 2018.10.23 14:42

(/u/5de885cec043)

大佬你竟然可以把RPC讲的这么好，棒棒棒

赞  回复





帅到自然醒呢 (/u/dfeea04c19e3)

19楼 · 2018.10.19 19:48

(/u/dfeea04c19e3)  
大佬，真棒！

赞 回复



今生愿为一个橙子 (/u/5e9a8c5923b9)

18楼 · 2018.09.27 23:12

(/u/5e9a8c5923b9)  
感谢大佬分享，最后两个链接失效了好像

赞 回复

柳树之 (/u/86696f09d988)：Dubbo加入apache后改版了，连以前的链接都换了，差评。。。新的链接在这<https://dubbo.incubator.apache.org/en-us/docs/user/quick-start.html>  
(<https://dubbo.incubator.apache.org/en-us/docs/user/quick-start.html>)

2018.10.08 13:58 回复

添加新评论



SoyaDokio (/u/b507ca3c8a91)

17楼 · 2018.09.22 02:48

(/u/b507ca3c8a91)  
这教程写得简单又不简单。

总结来说当担得起“深入浅出、通俗易懂”这八个字。从这篇文章来看，作者对RPC这套系统是有深入了解研究的。

BTW 这篇文章使我得以初窥RPC之门径，谢谢！

赞 回复



安安静静的小码农 (/u/baca54319b0b)

16楼 · 2018.09.12 09:40

(/u/baca54319b0b)  
好赞！！

赞 回复



光光不光 (/u/18a283e61680)

15楼 · 2018.09.06 16:56

(/u/18a283e61680)  
加油啊大佬

赞 回复

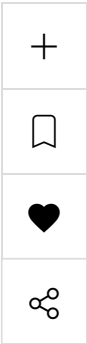



董圣均 (/u/ae7498b116a4)

14楼 · 2018.08.23 22:40


(/u/ae7498b116a4)  
good

赞 回复




 **kyrie233** (/u/d3763feecb1)  
13楼 · 2018.08.06 08:58  
(/u/d3763feecb1)  
大神继续呀！看完意犹未尽！


1人赞  回复

 **coolcalf** (/u/84e1a00648ef)  
12楼 · 2018.06.29 14:32  
(/u/84e1a00648ef)  
Wcf飘过


赞  回复

 **孟瑶123** (/u/f2652b359edb)  
11楼 · 2018.06.17 00:13  
(/u/f2652b359edb)  
很棒，后面等待你的完善版RPC

赞  回复

 **墨迹大王** (/u/bafee9409c33)  
10楼 · 2018.06.06 19:46  
(/u/bafee9409c33)  
看完了收货很多啊，也真的是感谢大佬写出这么通俗易懂的文章

赞  回复

 **Maxf\_aace** (/u/16feb5de3603)  
9楼 · 2018.06.06 15:43  
(/u/16feb5de3603)  
大佬写的东西，真的很很深入浅出！！  
期待后续！！

1人赞  回复


1


2

下一页

被以下专题收入，发现更多相似内容

+ 收入我的专题

 **Beautif...** (/c/ed1936817d67?utm\_source=desktop&utm\_medium=notes-included-collection)

 **@IT·互联网** (/c/V2CqjW?utm\_source=desktop&utm\_medium=notes-included-collection)





程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)



程序猿的进阶屋 (/c/eb15bc551b87?utm\_source=desktop&utm\_medium=notes-included-collection)



技术干货 (/c/38d96caffb2f?utm\_source=desktop&utm\_medium=notes-included-collection)



自己喜欢的技术文 (/c/19e3e163b2c5?utm\_source=desktop&utm\_medium=notes-included-collection)



tech (/c/abfb6ebe5c44?utm\_source=desktop&utm\_medium=notes-included-collection)

展开更多 ∨

(/p/5a1555add57b?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommendation  
**Nginx Https服务 + 反向代理+负载均衡 (/p/5a1555add57b?utm\_campaign=...**

1、Nginx Https服务 为了实现前后端分离的部署，希望实现如下的调用：比如：  
https://api.icheesedu.com/learn/search 提供接口给App端调用。比如：...



AKyS佐毅 (/u/4f07086fa936?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation

**用Python实现每秒处理120万次 HTTP 请求，你敢信？这已成为实现 (/p/b0...**

很多公司都在为了提升程序的执行性能和降低服务器的运营成本，而放弃 Python 去选择其它编程语言，其实这样做并不是必须，因为 Python 完全可以胜任这些任务。Python 社区最近做了大量关于性能的优化。...



妄心xyx (/u/906f252b4c34?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation

**接口测试-Java代码实现接口请求并封装 (/p/10f0aef2b184?utm\_campaign=...**

前言：在接口测试和Java开发中对接口请求方法进行封装都非常有必要，无论是在我们接口测试的时候还是在开发自测，以及调用某些第三方接口时，都能为我们调用和调试接口提供便捷；Java实现对http请求的...



蜗小粮 (/u/e02af7dce116?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation

**分布式服务接口的幂等性如何设计（比如不能重复扣款）？ (/p/54d85a9496...**

面试题 分布式服务接口的幂等性如何设计（比如不能重复扣款）？面试官心理分析 从这个问题开始，面试官就已经进入了实际的生产问题的面试了。一个分布式系统中的某个接口，该如何保证幂等性？这个事儿...



kevin0016 (/u/510cb159650c?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation




(/p/de272f28482d?

属性	描述
authorConfig	设置自述文件的选项配置
configScope	配置配置范围, 包括默认配置
default	设置默认值及关联的函数
default	设置默认值
dev	设置开发环境配置
envName	设置具体环境名称
health	设置应用健康指标
info	设置应用信息
imagePath	设置应用图标路径
instance	设置应用基本属性
metricsName	设置具体指标
pluginDev	应用应用
trace	设置应用追踪信息

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**一年了，我们都用 Spring Cloud 干了啥？ (/p/de272f28482d?utm\_campa...**

一、微服务构建框架 Spring Boot 配置 Spring Boot 是一个在 Spring 的基础上面做了很多简化的框架。首先得益于它习惯于配置的设计理念，所以从整个启动的容易的难度上来讲要简单非常多。它有一个 config ...

 荒城9510 (/u/fea24bbccac1?


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/ff59781653f2?

— 新手七日特训 —	
7月10日	王冠一《挑战无极限》
7月11日	张彦红《微商新人定位以及精装修》
7月12日	王佳《自动吸引力成交朋友圈打造秘籍》
7月13日	李连姬《美丽有约一美容季》
7月14日	孟凌亦《如何呈现你的美》
7月15日	艾静宇《绝密爆粉秘籍及客户精准化管理》
7月16日	何效欢《百分百成交术秘籍》

utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**付爱宝特种兵第九天成长日记 (/p/ff59781653f2?utm\_campaign=maleskin...**

每天时间充分感觉完全不够用，天天时间太紧了，每一天都恨不得一天当两天过，感恩付爱宝给我们创立此次特种兵训练，同时也感恩系统为我们的新代理不断创建新兵特训营，让我们每一个人都可以不同的成长...

 艾哲宇 (/u/9b05f5b9bca7?


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend

(/p/1609cc744c54?



utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**媒体眼中的枣庄 | 山东省滕州市畜牧业联合体的调查（1989年） (/p/1609cc...**

《人民日报》1989年2月5日 第2版 产供销一体化的有益尝试 ——山东省滕州市畜牧业联合体的调查 1981年开始组建的滕州市畜牧业产供销联合体，是在国家与农民之间的一种中间性服务组织。它囊括了不同所有...

 枣庄档案 (/u/579b8517a3e3?


utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommend


(/p/1957252f36e9?





utm\_campaign=maleskine&utm\_content=note&utm\_medium=seo\_notes&utm\_source=recommend  
**2017年高考志愿大讲堂在山东大学启动 (/p/1957252f36e9?utm\_campaign=...**


高考结束后，考生与家长却要在有限的十天内匆忙在2000多所院校与500多个专业之间挑选一所大学和一个专业，既没有太多的时间了解这么多的大学，又没有时间去剖析自己适合考什么专业，而且对于高考录取...









 老满说高考 (/u/839fd0887b17?)

utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommenc

+

