态牛 TECH NEO

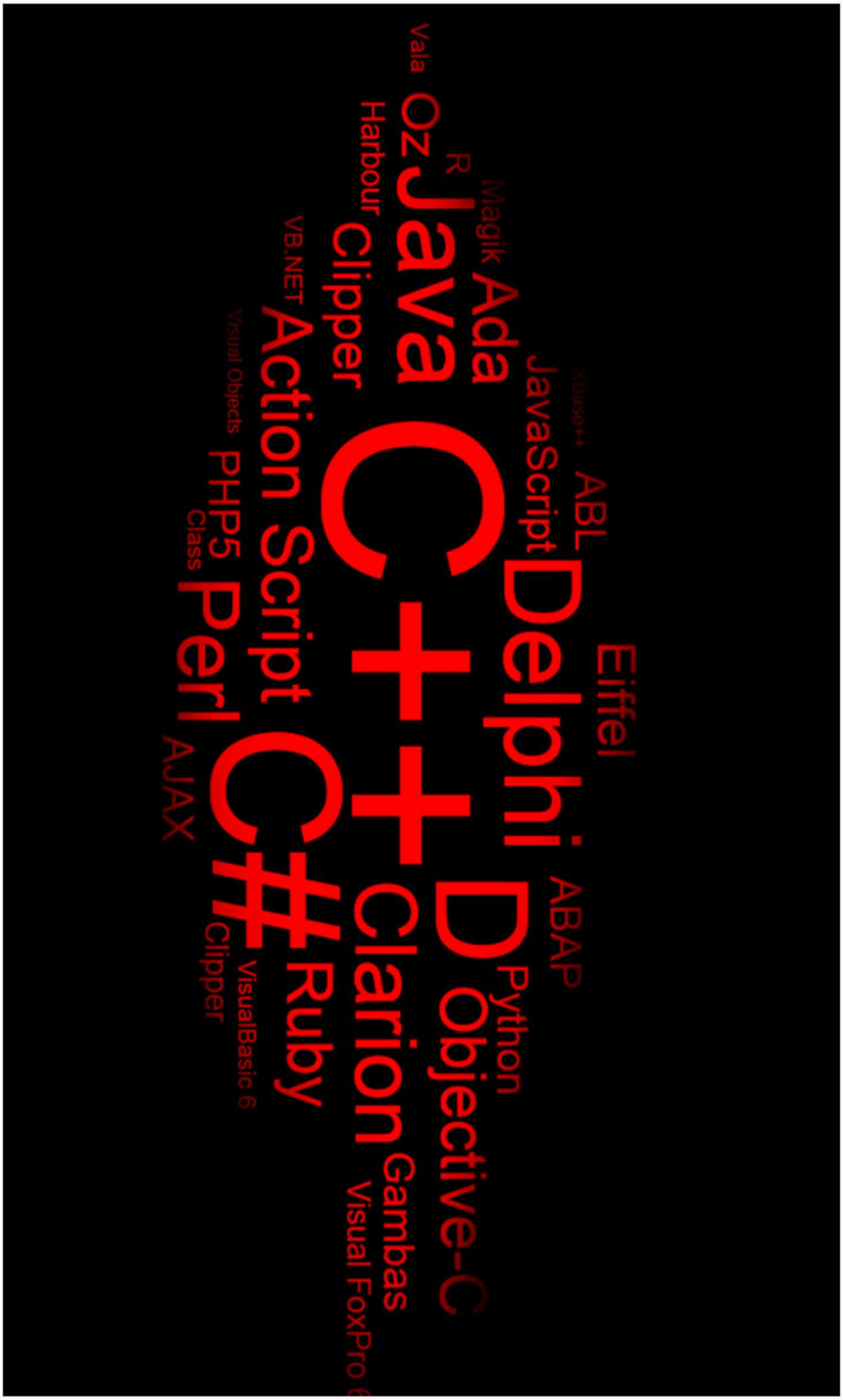
读我懂你，做千万开发者的选择

出版日期:2017/08
总期第37期
51CTO 网+

P4>>
探城-精选技术内容
探索互联网企业的核心技术

P55>>
坐诊-为研发难点把脉
看一线专家帮您解决技术难题

P38>>
挨踢部落故事汇系列
程序猿内心真实的职场故事



P17>>
2017年十大顶级编程语言新排名

P04>>
无需太过介意编程语言选择的四个理由

P06>>
我用Python爬了王者荣耀盒子

P18>>
阿里8年资深技术专家谈企业级互联网架构的演进之路

P15>>
作为大数据时代的DBA如何防止自己被淘汰

P33>>
用Sidecar安全代理保护Jaeger用户界面

P49>>
剖析链式存储栈与代码实现

态牛

出版方

51CTO网+

责任编辑

Liam

封面设计

冯海舰

联系方式

邮箱：linss@51cto.com

电话：010-68476606-8123

出版日期

2017年09月10日

投稿方式

linss@51cto.com

欢迎加入51CTO挨踢部落群

QQ群：312724475

51CTO.com 技术成就梦想

51CTO开发者交流群②群

扫一扫二维码，加入该群。

目录

2017年08月

总第37期

探城

04 无需太过介意编程语言选择的四个理由

06 我用Python爬了《王者荣耀盒子》，发现王者荣耀最强套路！

15 作为大数据时代的DBA，如何防止自己被淘汰

18 阿里8年资深技术专家谈企业级互联网架构的演进之路

22 区块链技术开源，是什么支撑BCOS的这种想法？

25 每秒高达千万分发，如何应对直播互动平台中海量消息挑战？

译文精选

28 七个聊天机器人构建平台：轻松打造自己的机器人

30 提升网站转化率的Web设计忠告

17 2017年十大顶级编程语言新排名

32 AWS披露面向Amazon S3的AI监控方案

33 用Sidecar安全代理保护Jaeger用户界面

挨踢故事汇

38 从逃离北上广到逃回北上广

42 一路前行，兴趣所至

46 扎根技术圈归属感爆棚

49 剖析链式存储栈与代码实现

52 全栈攻城狮之葵花宝典

技术坐诊

55 挨踢部落技术坐诊 · 精编篇

无需太介意编程语言选择的四个理由

来源：51CTO

编译：核子可乐

今天给大家推荐的内容包括：无需太过介意编程语言选择的四个理由、八种方式让Android成为强大生产力工具、值得在Twitter上关注的十位高影响力AI与机器学习专家和WannaCry勒索软件编写者犯下的那些低级失误等。

一、无需太过介意编程语言选择的四个理由

原文标题：

4 Reasons Why You Should Not Bother Too Much About Which Programming Language to Choose

每一位程序员都面临着这样的两难困境，即不知该学习哪种语言以高效成功地完成自身工作。编程语言多种多样，无论是新人还是老鸟都曾经一次又一次提出这个问题。在今天的文章中，我们将就此提出观点——语言本身并不重要。

1. 编程是一种概念

2. 编程的本质在于沟通

3. 没有哪种编程语言适合一切场景

4. 编程领域正在快速发展

二、八种方式让Android成为强大生产力工具

原文标题：8 ways to turn Android into a productivity powerhouse

毫无疑问，如今的Android已经成为出货量最高的智能手机类型，且在全球移动市场上占据三分之二以上份额。作为我们身边的智能中枢，如何才能让Android真正成为强大的生产力工具?在今天的文章中，我们将分享八种切实可行的方式，希望能够为朋友们的日常工作带来福音。

1. 优化主屏幕

2. 使用Gboard应用提升输入速度

3. 使用备忘录应用

4. 控制通知机制

5. 在应用间快速切换

6. 屏幕拆分

7. 运用Chrome的跨平台能力

8. 让Android本身更具效率

三、值得在Twitter上关注的十位高影响力AI与机器学习专家

原文标题：10 Influential AI and Machine Learning Experts to Follow on Twitter

值得在Twitter上关注的十位高影响力AI与机器学习专家

AI、机器学习、自然语言处理以及其它先进分析技术共同构建起了一套庞大的生态系统，可帮助企业从中真正实现大数据价值。那么，作为从业者，我们该如何把握时代的脉搏、了解技术发展趋势?下面，我们将介绍十位值得关注的高影响力AI与机器学习专家，他们的心得也许能够为各位朋友带来启示。

1. Josh Bloom

2. Kirk Borne

3. Erik Brynjolfsson

4. Rumman Chowdhury

5. Pedro Domingos

6. Tamara Dull

7. Jodi Goldstein

8. Fei-Fei Li

9. Cathy O’Neil

10. Monica Rogati

四、WannaCry勒索软件编写者犯下的那些低级失误

原文标题:

Amateur Mistakes by WannaCry Ransomware Hackers Every Developer Should Know

过去几个月以来，WannaCry已经在世界范围内掀起勒索软件攻击的新高潮，并导致成千上万用户沦为受害者——包括医院、政府机关以及电信运营商等等。尽管关于此轮攻击仍有不少未解之谜，但正是由于黑客方经验并不丰富，反而令安全专家陷入困惑。下面，我们将对WannaCry中的低级失误作出说明。

1. 文件擦除逻辑中存在错误

2. 文件存在于系统驱动器内

3. 存在只读取文件处理错误

■

来源：51CTO
作者：催家华

我用Python爬了《王者荣耀盒子》，发现王者荣耀最强套路！

程序员的鄙视链，您听说过吗?话说：老婆漂亮的程序员，鄙视老婆不漂亮的程序员;有老婆的程序员，鄙视没有老婆的程序员;没有老婆有女朋友的程序员，鄙视单身狗。

在单身狗之间，才有语言、编辑器和操作系统的互相鄙视。不知为何，看到这句话的时候，我的内心阵阵绞痛。、

现在看看，这游戏也是如此。玩星际的，鄙视玩 Dota 的;玩 Dota 的，鄙视玩 LOL 的;玩 LOL 的，鄙视玩王者荣耀的。

虽说王者荣耀处于游戏鄙视链的低端，但是不得不说，它确实有自己独到的魅力，正因为它降低了玩家的门槛，才有了如今的全民王者的热潮。你永远不知道自己匹配的队友和对手是老人，还是小孩。或者换种说法，是人，还是动物?

毫无疑问，王者荣耀从全民热度等诸多表现上看，是非常成功的游戏。可谓男女老少通吃，本文不讨论到底是女大学生坑，还是小学生坑，这样高难度的问题。咱玩点简单的，先看一款王者荣耀神器-王者荣耀盒子。

《王者荣耀盒子》是专门为《王者荣耀》玩家量身打造的一款攻略应用，可谓上分必备神器，这里有职业选手教你英雄出装、铭文搭配和各种对线团战技巧，同时它也整合最全的游戏资讯以及游戏实时动态。先看看它长什么样：

APP 下载地址：http://www.eoemarket.com/soft/870210.html

本文内容就是爬取《王者荣耀盒子》这款 APP 上的信息，废话不多说，直接开始实战!

运行平台： Windows

Python版本： Python3.x

IDE： Sublime text3

什么是 Fiddler?

Fiddler 是位于客户端和服务器端的 HTTP 代理，也是目前最常用的 HTTP 抓包工具之一 。

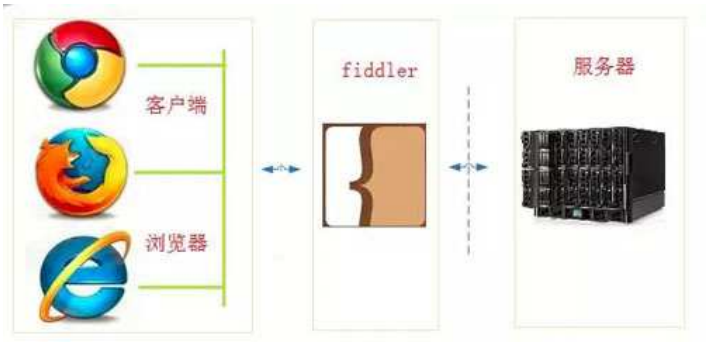
它能够记录客户端和服务器之间的所有 HTTP 请求，可以针对特定的 HTTP 请求，分析请求数据、设置断点、调试 Web 应用、修改请求的数据，甚至可以修改服务器返回的数据，功能非常强大，是 Web 调试的利器。

既然是代理，也就是说：客户

端的所有请求都要先经过 Fiddler，然后转发到相应的服务器，反之，服务器端的所有响应，也都会先经过 Fiddler 然后发送到客户端。

基于这个原因，Fiddler 支持所有可以设置 HTTP 代理为 127.0.0.1:8888 的浏览器和应用程序。

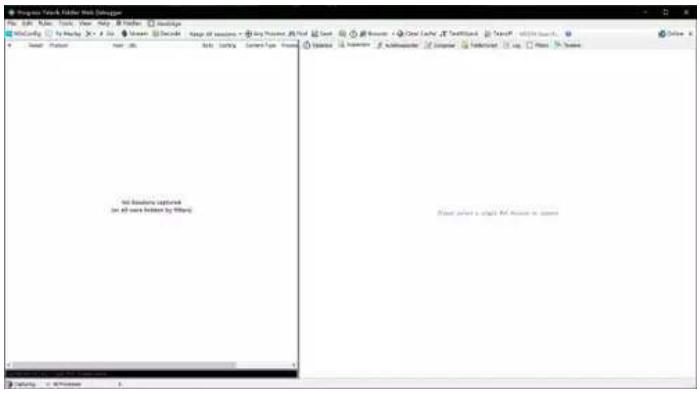
使用了 Fiddler 之后，Web 客户端和服务器的请求如下所示：



利用可以设置代理的这个特点，我们就可以对手机 APP 进行抓包了。怎么设置?不急不急，让我先把 Fiddler 安装上吧!

Fiddler 下载地址：http://fiddler2.com/

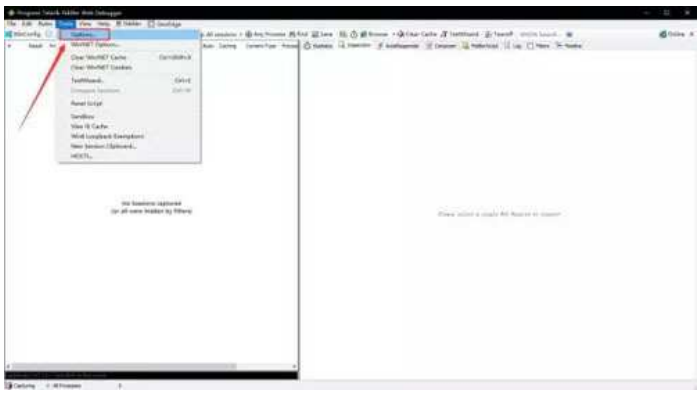
傻瓜式安装，一键到底。Fiddler 软件界面如图所示：



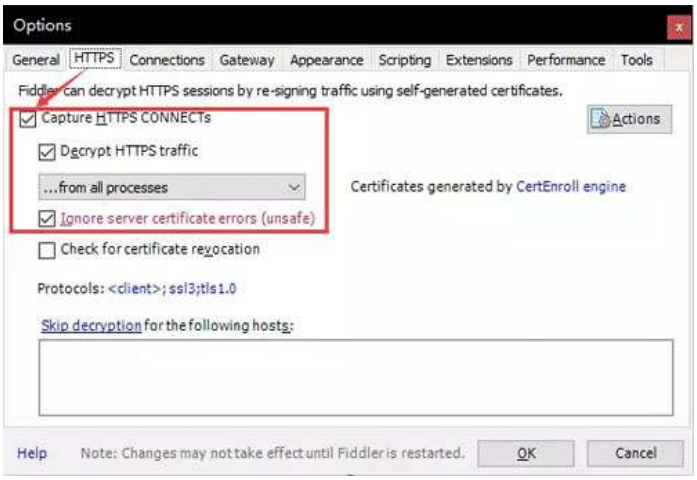
手机 APP 抓包设置

Fiddler 设置

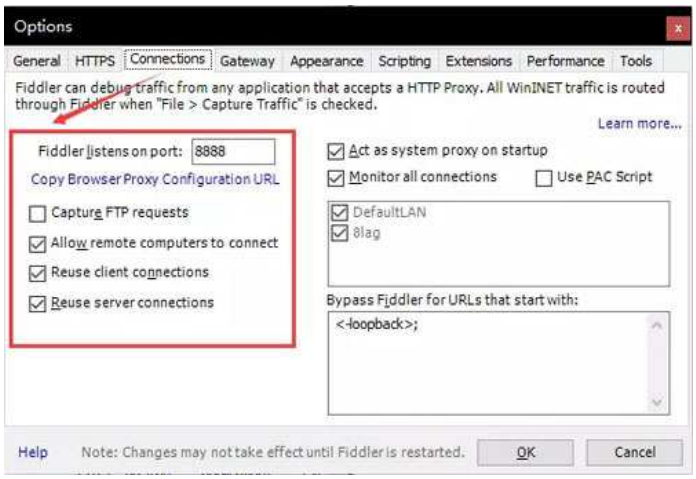
打开 Fiddler 软件，打开工具的设置。(Fiddler 软件菜单栏：Tools->Options)



在 HTTPS 中设置如下：



在 Connections 中设置如下，这里使用默认 8888 端口，当然也可以自己更改，但是注意不要与已经使用的端口冲突：



安全证书下载

在电脑浏览器中输入地址：

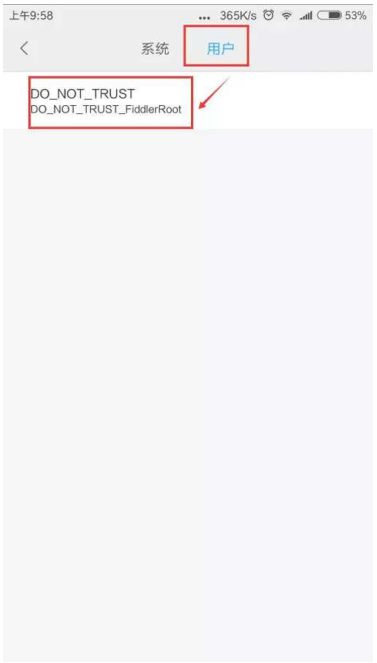
http://localhost:8888/，点击 FiddlerRoot certificate，下载安全证书：

安全证书安装

证书是需要在手机上进行安装的，这样在电脑 Fiddler 软件抓包的时候，手机使用电脑的网卡上网才不会报错。

将下载好的 FiddlerRoot.cer 安装证书拷贝到手机中，然后进行证书安装。以小米 5 手机为例：设置(Settings)->(系统和设备中的)更多设置->系统安全->从存储设备安装：

然后找到拷贝的 FiddlerRoot.cer 进行安装即可。安装好之后，可以在信任的凭证中找到我们已经安装好的安全证书，在用户中可以看到证书如下：

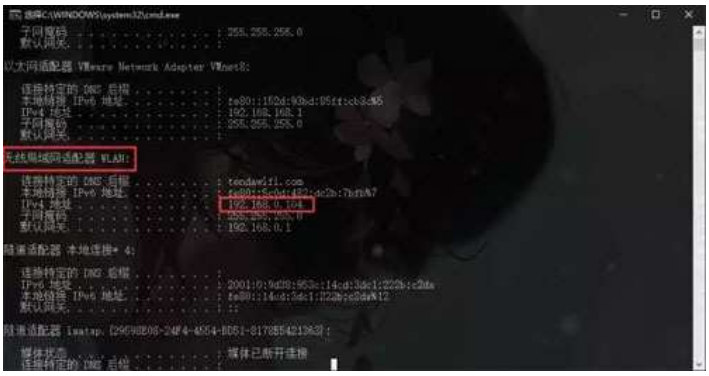


局域网设置

想要使用 Fiddler 进行手机抓包，首先要确保手机和电脑的网络在一个内网中，可以让电脑和手机都连接同一个路由器。

当然，也可以让电脑开放 WiFi 热点，手机连入。这里，我使用的方法是，让手机和电脑同时连入一个路由器中。最后，让手机使用电脑的代理 IP 进行上网。

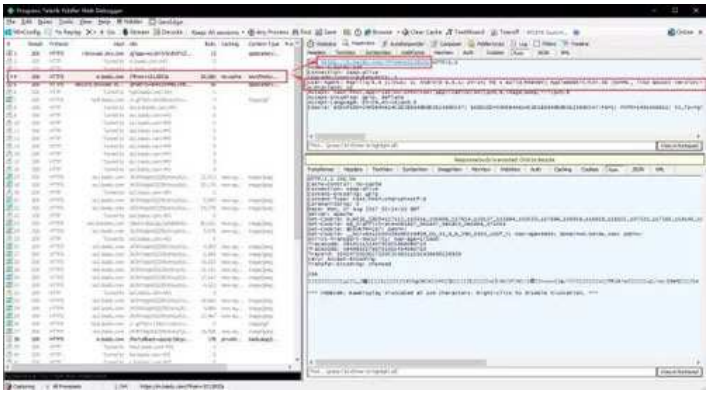
首先，查看电脑的 IP 地址，在 cmd 中使用命令 ipconfig 查看电脑 IP 地址。找到无线局域网 WLAN 的 IPv4 地址，记下此地址。



在手机上，点击连接的 WiFi 进行网络修改，添加代理。进行手动设置，主机名即为上图中找到的 IP 地址，端口号即为 Fiddler 设置中的端口号 8888：

Fiddler 手机抓包测试

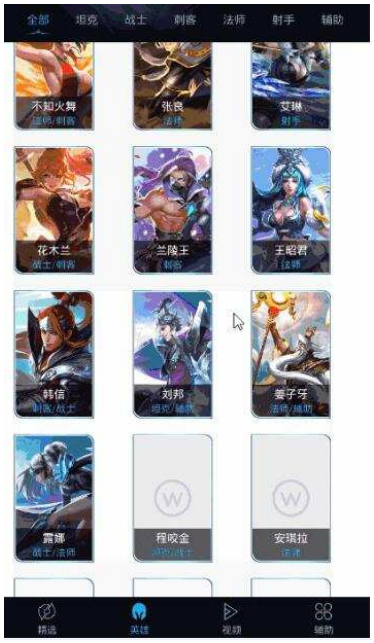
上述步骤都设置完成之后，用手机浏览器打开百度首页，我们就可以顺利抓包了，截图如下：



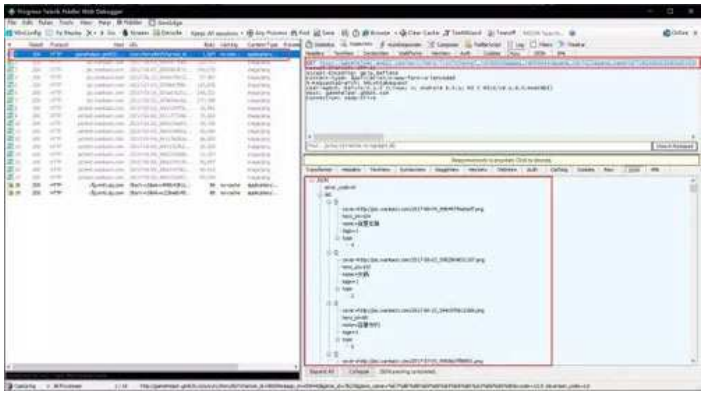
实战走起

下载英雄图片

先来个简单的例子热热身，我想将《英雄联盟盒子》中的英雄图片下载下来，该如何操作？



先使用 Fiddler 抓包看一下，在手机 APP《英雄联盟盒子》中的菜单中点击英雄，电脑 Fiddler 就会抓包如下：



可以看到，GET 请求的 url 地址，和返回的 JSON 格式的信息。那么编写代码如下：

```
import requests
if __name__ == '__main__':
    headers = {'Accept-Charset': 'UTF-8',
               'Accept-Encoding': 'gzip,deflate',
               'User-Agent': 'Dalvik/2.1.0 (Linux; U; Android 6.0.1; MI 5 MIUI/V8.1.6.0.MAACNDI)',
```

```
6.                ‘X-Requested-With’:
‘XMLHttpRequest’,
7.                ‘Content-type’: ‘application/
x-www-form-urlencoded’,
8.                ‘Connection’: ‘Keep-Alive’,
9.                ‘Host’: ‘gamehelper.gm825.
com’}
10.       heros_url = “http://gamehelper.gm825.
com/wzry/hero/list?channel_id=90009a&app_
id=h9044j&game_id=7622&game_name=%E7%8E%8B%E8%
80%85%E8%8D%A3%E8%80%80&vcode=12.0.3&versi
on_code=1203&cuid=2654CC14D2D3894DBF5808264AE2
DAD7&ovr=6.0.1&device=Xiaomi_MI+5&net_
type=1&client_id=1Yfyt44QSqu7PcVdDduBYQ%3D%3D&
info_ms=fBzJ%2BCu4ZDAtl4CyHuZ%2FJQ%3D%3D&in
fo_ma=XshbgIgi0V1HxXTqixI%2BKbgXtNtOP0%2Fn1WZt
MWRWj5o%3D&mno=0&info_la=9AChHTMC3uW%2BfY8%2BC
FhcFw%3D%3D&info_ci=9AChHTMC3uW%2BfY8%2BCFhcFw
%3D%3D&mcc=0&clientversion=&bssid=VY%2BeiuZRJ%
2FwaXmoLLVUrMODX1ZTf%2F2dzsWn2AOEM0I4%3D&
os_level=23&os_id=dc451556fc0eeadb&resolution=
1080_1920&dpi=480&client_ip=192.168.0.198&pdun
id=a83d20d8”
11.       req = requests.get(url = heros_url,
headers = headers).json()
12.       print(req[‘list’])
```

从代码运行结果中可以看到，百里玄策的英雄 ID 为 104，他的图片存放地址为

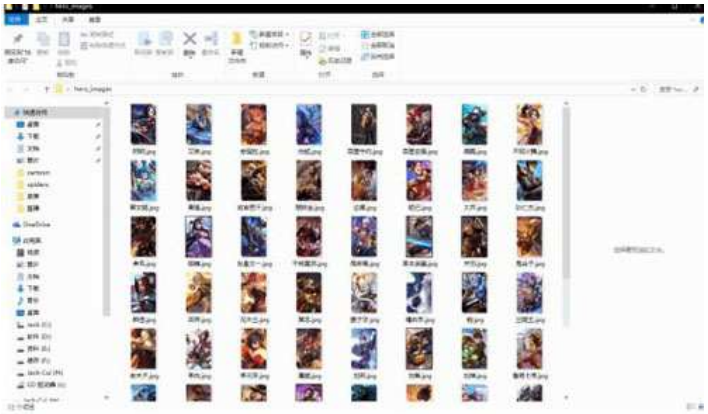
http://pic.wankacn.com/2017-08-04_59844776a6adf.png:

知道了这些信息，我们就可以将这些英雄的图片进行下载了，编写代码如下：

```
1.    from urllib.request import urlretrieve
2.    import requests
3.    import os
4.    “”””
5.    函数说明:下载《英雄联盟盒子》中的英雄图片
6.    Parameters:
7.        heros_url - GET请求地址，通过Fiddler抓
包获取
8.        header - header信息
9.    Returns:
10.       无
11.   Author:
12.       Jack Cui
13.   Blog:
14.       http://blog.csdn.net/c406495762
15.   Modify:
16.       2017-08-07
17.   “”””
18.   def hero_imgs_download(heros_url,header):
19.       req = requests.get(url = heros_url,
headers = header).json()
20.       hero_num = len(req[‘list’])
21.       print(‘一共有%d个英雄’ % hero_num)
22.       hero_images_path = ‘hero_images’
23.       for each_hero in req[‘list’]:
24.           hero_photo_url = each_
hero[‘cover’]
25.           hero_name = each_hero[‘name’] +
‘.jpg’
26.           filename = hero_images_path + ‘/’
+ hero_name
27.           if hero_images_path not in os.
listdir():
28.               os.makedirs(hero_images_path)
```

```
29.         urlretrieve(url = hero_photo_url, filename = filename)
30.  if __name__ == '__main__':
31.      headers = {'Accept-Charset': 'UTF-8',
32.              'Accept-Encoding': 'gzip,deflate',
33.              'User-Agent': 'Dalvik/2.1.0 (Linux; U; Android 6.0.1; MI 5 MIUI/V8.1.6.0.MAACNDI)',
34.              'X-Requested-With': 'XMLHttpRequest',
35.              'Content-type': 'application/x-www-form-urlencoded',
36.              'Connection': 'Keep-Alive',
37.              'Host': 'gamehelper.gm825.com'}
38.      heros_url = "http://gamehelper.gm825.com/wzry/hero/list?channel_id=90009a&app_id=h9044j&game_id=7622&game_name=%E7%8E%8B%E8%80%85%E8%8D%A3%E8%80%80&vcode=12.0.3&version_code=1203&cuid=2654CC14D2D3894DBF5808264AE2DAD7&ovr=6.0.1&device=Xiaomi_MI+5&net_type=1&client_id=1Yfyt44QSqu7PcVdDduBYQ%3D%3D&info_ms=fBzJ%2BCu4ZDAtl4CyHuZ%2FJQ%3D%3D&info_ma=XshbgIgi0V1HxXTqixI%2BKbgXtNtOP0%2Fn1WZtMWRWj5o%3D&mno=0&info_la=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&info_ci=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&mcc=0&clientversion=&bssid=VY%2BeiuZRJ%2FwaXmoLLVUrMODX1ZTf%2F2dzsWn2AOEM0I4%3D&os_level=23&os_id=dc451556fc0eeadb&resolution=1080_1920&dpi=480&client_ip=192.168.0.198&pdunid=a83d20d8"
39.      hero_imgs_download(heros_url,headers)
```

运行上述代码，看下结果，72 个英雄的图片，分分钟搞定：

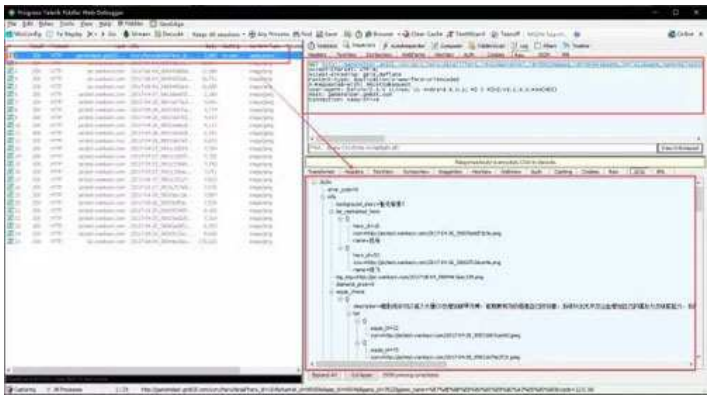


是不是很简单？接下来，再玩一个加点难度的。

英雄推荐出装查询助手

点击百里玄策这个英雄，可以看到，里面有他的简介，包括技能介绍，以及推荐出装等。那么，我们就了利用这个做一个自己的英雄出装查询小助手吧。

手机点击百里玄策这个英雄，可以在电脑 Fiddler 看到如下抓包内容：



我们看下 GET 请求的 URL:

```
http://gamehelper.gm825.com/wzry/hero/detail?hero_id=104&channel_id=90009a&app_id=h9044j&game_id=7622&game_name=%E7%8E%8B%E8%80%85%E8%8D%A3%E8%80%80&vcode=12.0.3&version_code=1203&cuid=2654CC14D2D3894DBF5808264AE2DAD7&ovr=6.0.1&device=Xiaomi_MI+5&net_type=1&client_id=1Yfyt44QSqu7PcVdDduBYQ%3D%3D&info_ms=fBzJ%2BCu4ZDAtl4CyHuZ%2FJQ%3D%3D&info_ma=XshbgIgi0V1HxXTqixI%2BKbgXtNtOP0%2Fn1WZtMWRWj5o%3D&mno=0&info_la=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&info_ci=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&mcc=0&clientversion=&bssid=VY%2BeiuZRJ%2FwaXmoLLVUrMODX1ZTf%2F2dzsWn2AOEM0I4%3D&os_level=23&os_id=dc451556fc0eeadb&resolution=1080_1920&dpi=480&client_ip=192.168.0.198&pdunid=a83d20d8
```

很有特点发现了吗?在 url 的 heroid 为 104，在上个小实例中，我们发现，这个 104 正好是英雄的 ID，那么是不是可以通过设置这个 ID 来访问不同的英雄界面呢?

测试一下答案就出来了，我们将 ID 改为 103，也就是女娲的 heroid，编写代码如下:

```
import requests
if __name__ == '__main__':
    headers = {'Accept-Charset': 'UTF-8',
    'Accept-Encoding': 'gzip,deflate',
    'User-Agent': 'Dalvik/2.1.0 (Linux; U; Android 6.0.1; MI 5 MIUI/V8.1.6.0.MAACNDI)',
    'X-Requested-With': 'XMLHttpRequest',
    'Content-type': 'application/x-www-form-urlencoded',
    'Connection': 'Keep-Alive',
    'Host': 'gamehelper.gm825.com'}
    hero_url = "http://gamehelper.gm825.com/wzry/hero/detail?hero_id={}&channel_id=90009a&app_id=h9044j&game_id=7622&game_name=%E7%8E%8B%E8%80%85%E8%8D%A3%E8%80%80&vcode=12.0.3&version_code=1203&cuid=2654CC14D2D3894DBF5808264AE2DAD7&ovr=6.0.1&device=Xiaomi_MI+5&net_type=1&client_id=1Yfyt44QSqu7PcVdDduBYQ%3D%3D&info_ms=fBzJ%2BCu4ZDAtl4CyHuZ%2FJQ%3D%3D&info_ma=XshbgIgi0V1HxXTqixI%2BKbgXtNtOP0%2Fn1WZtMWRWj5o%3D&mno=0&info_la=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&info_ci=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&mcc=0&clientversion=&bssid=VY%2BeiuZRJ%2FwaXmoLLVUrMODX1ZTf%2F2dzsWn2AOEM0I4%3D&os_level=23&os_id=dc451556fc0eeadb&resolution=1080_1920&dpi=480&client_ip=192.168.0.198&pdunid=a83d20d8".format('103')
    req = requests.get(url = hero_url, headers = headers).json()
    print(req['info'])
```

运行上述代码，我们可以看到，打印的信息确实是英雄女娲的信息，返回的信息显示，这个英雄还没有上线。

推荐出装保存在 equip_choice 中，可以看到这里没有给出装备的名字，只有装备的 equip_

id，那么在写推荐出装小程序之前，我们需要获取所有装备的 ID。

怎样获取武器的信息?抓包方法同上，点击装备大全:

通过 Fiddler 抓包信息，编写代码如下:

```
import requests
if __name__ == '__main__':
    headers = {'Accept-Charset': 'UTF-8',
               'Accept-Encoding': 'gzip,deflate',
               'User-Agent': 'Dalvik/2.1.0 (Linux; U; Android 6.0.1; MI 5 MIUI/V8.1.6.0.MAACNDI)',
               'X-Requested-With': 'XMLHttpRequest',
               'Content-type': 'application/x-www-form-urlencoded',
               'Connection': 'Keep-Alive',
               'Host': 'gamehelper.gm825.com'}
    weapon_url = "http://gamehelper.gm825.com/wzry/equip/list?channel_id=90009a&app_id=h9044j&game_id=7622&game_name=%E7%8E%8B%E8%80%85%E8%8D%A3%E8%80%80&vcode=12.0.3&version_code=1203&cuid=2654CC14D2D3894DBF5808264AE2DAD7&ovr=6.0.1&device=Xiaomi_MI+5&net_type=1&client_id=1Yfyt44QSqu7PcVdDduBYQ%3D%3D&info_ms=fBzJ%2BCu4ZDAtl4CyHuZ%2FJQ%3D%3D&info_ma=XshbgIgi0V1HxXTqixI%2BKbgXtNtOP0%2Fn1WZtMWRWj5o%3D&mno=0&info_la=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&info_ci=9AChHTMC3uW%2BfY8%2BCFhcFw%3D%3D&mcc=0&clientversion=&bssid=VY%2BeiuZRJ%2FwaXmoLLVUrMODX1ZTf%2F2dzsWn2AOEM0I4%3D&os_level=23&os_id=dc451556fc0eeadb&resolution=1080_1920&dpi=480&client_ip=192.168.0.198&pdunid=a83d20d8"
    req = requests.get(url = weapon_url, headers = headers).json()
    print(req['list'])
```

这样我们就轻松得到了各个装备的信息:

到这里，我们就可以整合代码了，首先通过获

取每个英雄的 ID，然后根据每个英雄的 ID，再获得英雄的详细信息，包括推荐出装，最后通过推荐的装备 ID，找到装备的信息并打印出来。

根据实现效果，自己编写代码试一试如何？《王者荣耀》出装小助手，运行效果如图所示：

如果功能已经实现了，那么先恭喜一声！

上述功能代码获取地址：https://github.com/Jack-Cherish/python-spider/blob/master/hero.py，欢迎 Follow、Star!

总结

本文的例子对于入门手机 APP 内容的爬取有一定的帮助作用。趁着《王者荣耀盒子》APP 没有更新，很好爬，赶紧动手实践吧！

爬取 APP 的内容和爬取网页的思路是一样的，都是通过抓包进行分析。如有问题，请留言。如有错误，还望指正，谢谢！

催家华

知名博客博主

现就读于东北大学模式识别与智能系统专业。本科期间，曾担任学生会”科技创新中心主任”一职，负责组织各类科技竞赛相关活动。与此同时，热爱科技竞赛，曾于 2015 年获得第十届全国大学生“飞思卡尔”杯智能车竞赛全国一等奖。研究生期间，致力于机器学习，在知名博客已取得了不错的关注度。

■

作为大数据时代的DBA：如何防止自己被淘汰

来源：51CTO
作者：赵建春

一句话对比三大数据库软件

这个世界上Oracle、SQL Server和MySQL三大数据库软件呈三足鼎立之势，三者之间互有优缺点。网上能查到的内容我不再赘述，我把三者的优缺点对比总结成了三行文字：

Oracle：最贵，功能最多，支持平台数量较多，速度中等，开发和运维比较简便，最安全。

SQL Server：价格适中，功能比较丰富，支持平台单一，速度最慢，开发和运维最简便，比较安全。

MySQL：价格最低廉，功能最少，支持平台最多，速度最快，开发和运维最繁琐，最不安全。

DBA选择数据库的方法

1、按性能和安全性来选择

大型国企（银行）、传统企业上市公司（保险）的企业级应

用对数据的完整性和安全性要求最高，务必万无一失，因此安全性最佳、功能最齐全的Oracle是传统软件的首选。一般来说移动互联网领域的公司对数据的完整性和安全性的敏感程度并不太高，所以互联网行业的从业者可能有入行几年还没见过Oracle的，不过电商、金融类的互联网产品对安全性要求同样非常高的软件也应该选择Oracle作为数据库。

如果公司是非移动互联网行业的外企，那么一定要选择Orcale，一来是洋老板普遍财大气粗，二来是外企领导不像国内领导这么依赖对Windows系统友好的数据库软件。

2、按开发速度和运维难度来选择

一般国企和事业单位以及传统企业小公司的领导对软件（一般情况下只有展示网站，连ERP系统都是买现成的）的要求就是价格低、快点出结果；加上国内的传统的领导对Windows系统有说不清的迷恋；如此一来选择SQL Server能同时满足这三个要求，不过一般在这样的单位不会存在DBA这个岗位——让开发网站的程序员担任DBA的一部分职责（其实也就只有备份、恢复、作业管理等几样）。

3、按价格和速度因素来选择

移动互联网时代的创业公司并不需要太纠结数据的万无一失，但很在意成本，并且对数据访问速度要求很高。因此资金不太充裕、为了给移动端提供API的服务器选择MySQL是最明智的。事实上在互联网领域MySQL的市场占有率是最高的，大多数移动互联网公司都用MySQL作为数据库。

Oracle和SQL Server都是具备完善功能的数据库生态系统，而MySQL仅是一个速度快、兼容性好的关系型数据库软件。但好在MySQL是开源的，同时追求速度快和数据安全的公司（移动互联网领域的电商、金融公司）在资金充裕可以专人负责写一个自己的MySQL系统，满足系统的高可用和、高并发、高可扩的需求。

尽管MySQL的安全性一直为人所诟病，但经过多年改良，安全性已经超越了SQL Server。

DBA如何优化SQL

一个合格的DBA会明白无论选择Oracle、MySQL还是SQL Server，SQL语句的优化都是数据库优化的决定性因素，；而对数据库工具本身的优化只不过是细枝末节，别人的配置方式也很容易从网上查到。

SQLite是Android系统上运行的数据库软件，性能与三大数据库软件不可同日而语，况且在Java语言的IDE里编写SQL还没有自动提示，有时候Android程序员对SQL语句优化的要求比DBA还高，在SQL上吃过不少亏的我总结了一些SQL语句优化方案，在此抛砖引玉，并非外行指导内行：

（1）在FROM语句中包含多个表的情况下，把条目最少的表放在前面。比如表一记录某省人口数据，表二记录全国失学儿童数据，那么查询某省的失学儿童应该把表二放在前面。

（2）在WHERE语句中过滤掉最多数据的条件必须卸载前面。列如把性别写在生肖前面的效率是生肖写在性别前面的6倍。

（3）避免使用 SELECT * 因为这样会查找所有的列名，耗费更多的时间。

（4）在SQL语句的WHERE和JOIN部分中用到的所有字段上，应该加上索引。

（5）移除不必要的表，可以减少很多数据库的流程。

（6）尽管SQL是大小写不敏感的语言，规范的SQL语句务必大写。

DBA在大数据时代的发展前景

成为一个DBA表面上“仅”需要SQL语言和数据库基础管理的知识，所以造成了从业者良莠不齐，举一个极端的例子，我有一个留级两年的同学勉强拿到毕业证后就从事了DBA。然而理论跟实际相差甚远，模拟的环境根本达不到线上要求的多用户、高并发等压力，仅在课堂上和自学是不能成为一个合格的DBA的，合格的DBA必须在企业中的实战中走出来。一个合格的DBA必须掌握Linux基本命令和操作、数据库备份和恢复、SQL优化、数据库性能优化和至少一门脚本语言（如Python）。

移动互联网时代雨后春笋般诞生的创业公司给人一种DBA的角色性变弱的错觉，其实任何IT公司做大后都离不开DBA这个岗位。DBA的发展前景一直很不错，尤其在大数据的时代里，原本就一直存在的性能问题和安全问题会更加突出，这意味着企业对DBA人才的需求会越来越旺盛，同时这也意味着DBA会面临比以往更多样化的困难和更严峻的挑战。加上近年来的大规模集群和自动化运维，很多以前手工维护的工作现在都可以通过一系列自动工具来完成，尽管让只会增删改查的DBA看到了自己的可替代性，但也给对技术精益求精的DBA节约出了钻研新技术和新知识的时间，使他们更加专注于解决所前所未有的困难。

大数据时代是一个以数据驱动决策的时代，数据就是企业的生命，掌握公司命脉的DBA的岗位越来越重要，作用也越来越显著，待遇也越来越丰厚，发展前景也越来越明朗。

2017年十大顶级编程语言新排名

2017年十大顶级编程语言新排名、大数据技术三大发展趋势以及三项负面干扰因素、你不可错过的五款强大云生产力工具和七款酷炫物联网应用与相关开发心得等。

点击这里查看高清大图

编程的世界浩瀚无垠，开发者绝不可能单凭自身熟知的一款工具即走遍天下。然而，特定的语言、平台以及框架却确实能够在拉低上手门槛的同时，显著提升你的工作能力与实现效果。着眼于2017年，我们整理出以下编程语言榜单，希望能为大家的进一步学习与选择带来一点启示。

1. Python
2. C
3. Java
4. C++
5. C#
6. R
7. JavaScript
8. PHP
9. Go
10. Swift

■

来源：51CTO
作者：王晶昱

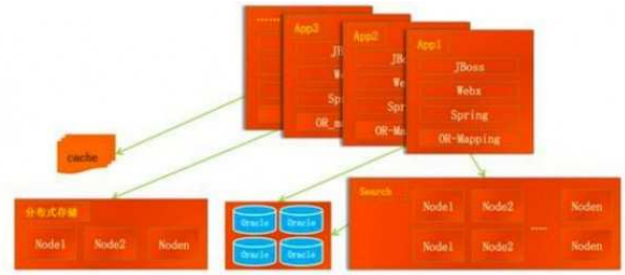
阿里8年资深技术专家谈企业级互联网架构的演进之路?

沈询，阿里巴巴中间件&稳定性平台资深技术专家，在淘宝工作八年间，主要负责的产品有淘宝分布式数据库（TDDL/DRDS）、分布式消息系统(Notify/ONS)等，故对整个分布式的互联网架构比较了解。本文分享围绕阿里技术架构演进及过程中遇到的问题与企业级信息系统架构的演进展开。

阿里技术架构演进及过程中遇到的问题

2003 年，淘宝最初的架构建设是采用 PHP，实践发现系统抗压能力相对薄弱。因为淘宝是一个企业级系统，所以选择了当时非常重要的企业级技术 JavaBean。之后把整个 Web 的容器、EJB 等整套体系引入淘宝，雇用有经验的工程师一起做架构。淘宝在技术方面也走过很长的路，在架构建设过程中，大家讨论最多的事情是如何划分模块。

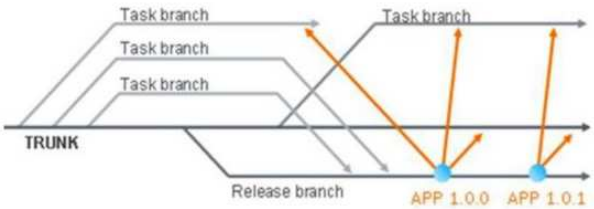
随着技术的不断发展，到 2006 年，淘宝技术又从 EJB 过渡到 Spring，如下图：



目前，这个产品已经开源，最上层采用的是 JBoss、中间采用 Webx，之后是 Spring、OR-Mapping，底层用到是 Oracle 数据库。用 Search 做搜索引擎，是因为当时收购雅虎，把雅虎的搜索引擎挂接到淘宝。像这样采用分布式的存储方式在现在看来很常见。

当时，业务不断高速发展，一些问题随之逐渐暴露出来。这里主要分享工程维护、人员变动、数据孤岛、数据集能力不足等问题。

工程维护与人员变动



随着业务不断壮大，技术团队的工程也会越来越多，源代码加速膨胀。多个工程之间，源代码冲突严重同时因为工作没有边界导致相互协同的成本不可估量。

假设出现项目完成，核心人员离职的情况，项目维护也会成为问题，当新人入职之后，学习老代码的难度也可想而知。

数据孤岛

数据孤岛是各个公司很普遍的问题，在那个时期，天猫还叫淘宝商城，不是基于淘宝，而是完全独立的一个组。

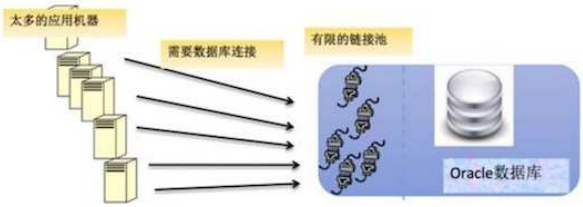
后期因为一些原因，想要把两个体系合并，却因为各自独立的业务体系、用户 ID、数据存储格式等等差异导致操作困难。且数据本身质量不高，做统一分析也有难度。

数据库能力达到上限

当时用的是小型机+Oracle，CPU90% 以上，每年宕机最少一次。这主要是因为有大量新业务写入，两周一次的频度，不断地有新 SQL 产出。

在新的 SQL 中，如出现一个慢 SQL，就会出现宕机。当时我们用的小型机重启一次需要 20 分钟，切换到异地也是 20 分钟。

关于连接数问题，如下图：



当时后端 Oracle 的连接池有限，约 8000 个左右，一旦超过就会出现问题。因为超过数量，链接占的内存会非常大，且连接数单点风险系统很高。

阿里面对 DBA 相关问题的应对方法

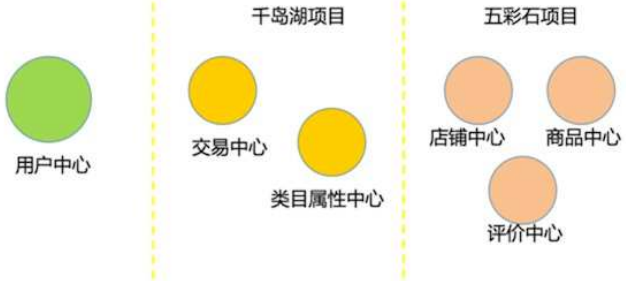
综上所述，当时阿里 DBA 面临维护人员很多，团队职责不清、数据无法共享，团队各自为战、小型机数据库压力过大，连接数单点风险系统很高等问题。

好在阿里那时正处于增长期，所以这时通过招聘一些技术大牛来解决问题。

基于 EDAS 进行服务化改造

针对阿里 DBA 遇到的问题，从硅谷请来的技术人用服务化的方式试着解决。当时在中国只有用友做过服务化，且效果不是很好，没有借鉴，只能谨慎小心的自己往前走。

如下图，是阿里以服务化方式将系统专业分工的三个关键战役。



用户中心服务化

选择用户中心的第一个是做服务化，因为用户中心是最小集合，最简单清楚，还因为确实有业务需求，也是想要验证这条服务化的理念是不是正确。

服务化之前的用户中心，有六个不一样的查询方法，看起来遍历的方式差不多，但可能某个参数不同，因为数据来自不同的团队。

服务化的原则是能不改不改，能简化简化，采用的传输方式是 HTTP。然而，这样做行不通，是因为除了服务化 HTTP，其他内容没有改变，就需要布设 Load Balance。

为了保证 Load Balance 尽可能稳定，所以选择硬件F5来配置。把前端进入的用户流量打到F5，额外在增加新VIP接口，请求通过F5转出去。

这里发现一个很严重的问题，就是每当用户登陆一次，出现一个节点，跳转一次流量就要增加一倍。但 F5 是很贵的设备，未来如果所有都变成服务化，用 F5 就不可行。

千岛湖项目

配置 F5 负载均衡行不通，换了另一种思路就是由集中的单点模式变成真正意义的分散模式。当时阿里把这样的方式叫软负载，做的是分散负载均衡的事情。

当做交易中心服务化时，必然要用事物相关的方式，来保证整个流程的稳定性、一致性，当时采用的是最终一致性的设计方法。之后，通过实践反复修改，优化，得到稳定的消息系统。

有了消息系统的研发经验，随后类目属性等中心也随之服务化，之所以叫千岛湖项目，是因为大家很辛苦，完成项目之后去千岛湖旅游。

五彩石项目

随着千岛湖项目完成，底层架构、中间件的稳定，之后要做的事情就是把庞大的系统全部一次性服务化。

恰逢此时，淘宝商城和淘宝需要合并，所以整个系统在那个时期进行了彻底的拆分，也就是淘宝 3.0。

之后再也没有出现 4.0，一直采用服务化的架构方式。

基于 DRDS 进行数据库分布式改造

DRDS 建设初衷是希望对 Oracle 进行数据解析，同步到 MySQL 中。当时大家觉得 Oracle 很稳定，整个系统不会丢数据，所以要把 Oracle 放在主机。

但也发现一个问题，一是小型机比较贵重需要在后端加入大量 PC 机，进行读写分离。还有就是看似稳定的 Oracle，在 Linux 环境中表现不是很好，尤其是两者还在兼容上存在很多问题。

如下，是传统数据库向分布式数据库的转化图：

最终选择用分布式的 MySQL 架构来解决问题，借鉴 Facebook 技术团队开发的开源项目 Flashcache 机制，为 MySQL 加速，完成整个业务的部署。在 Linux 环境下，MySQL 比 Oracle 更加稳定、运维成本也相对较低。

如上是做服务化中心带来的好处，显而易见。经过一段时间的改造之后，技术架构发生了很大变化，如下图：

在整个技术架构的最底层是 EDAS、DRDS、ONS 等组成的基础应用架构。电商、物流、移动IM、地理信息、医疗等都有自己的能力且都把能力进行开放，形成了IT共享业务架构层，用来支撑上层的业务。一旦业务需要合作，下层的技术可以快速响应。

例如，淘宝和滴滴从谈合作到项目顺利完成，只用了不到五天的时间。当时是一个内网打车软

件，用这个软件打车可实现通过公司账户去叫车，省去贴发票环节。

这件事情，真正的凸显出，技术可以帮助业务解决一些问题。虽然淘宝可能再也没有 4.0 版本，但现在每套应用系统都是为了未来而准备。

企业级信息系统架构的演进

通过对阿里技术发展历程的总结，我们发现这些经过实践得出来的技术架构对一部分企业可以起到借鉴的作用。所以规划成一个产品——企业级信息系统。

云计算时代，企业信息化演进不仅仅是把 IT 系统搬到云上，而是让业务与信息系统深度融合，改变业务运营和创新模式。

如下图，是企业级信息系统的演进过程：

把业务云化，从虚拟机模式转变成基于分布式的互联网架构进行重构，重构后给企业带来的主要的价值是原来的一些效率问题得以解决。所以说，互联网架构平台是企业云上演进的使能平台。

图为企业级互联网架构平台对应的下层架构：

最底层是基础框架，主要涉及 EDAS、MQ 和 DRDS 三大产品：

EDAS 主要职责是业务应用的编写和发布。

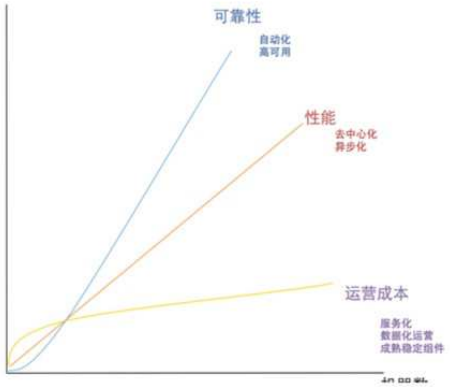
MQ 是在异步解耦的过程中，用来做消息的编辑和保证事物的一致性。

有大量的用户和数据后，由 DRDS 负责分散到各个应用中去。

三个产品加起来，从上到下，很好地支撑业务的编写、相互通信以及下层数据的建设。

CSB（能力开放平台）的主要职责是将阿里的 negligible 对外开放运营、保证内部数据的安全性和开放性，同时和现有的系统打通，进入到系统中去。云化业务的能力部分，是和客户一起设计构建。整套系统到最后的核心关键点是成本、稳定和效率。阿里在业务高速增长的这十几年实践中，积累了很多这三方面的经验。希望这些经验可以帮助一些企业少走弯路。

如下图，是企业级互联网架构的关键特征：



随着机器数量的增加，性能一定是线性增加、可靠性成指数型增长、运营成本要保持对数级上升。这些才是真正互联网架构的关键特征，如果系统不能很好的解决这些问题，它会是一个无法向上扩展的系统，那么就无法满足未来用户的增长需要。■

区块链技术开源，是什么支撑BCOS的这种想法？

来源：51CTO

作者：林师授

被大家经常提及的除了区块链的技术体系外，究竟它的核心价值是什么？是更好的性能、更方便的架构、更可靠的技术，还是其他的因素？关于这个问题，在2017年8月14号的区块链开源讨论会上，来自微众、万向、矩阵元的领导总结了一个非常到位的词，那就是“信任”。

区块链技术从2008年被首次提起到如今发展如火如荼，同时在互联网上也产出了大量不同行业深度的区块链体系文章介绍，从底层技术、网络结构、共识算法等做出了的很多深度的阐述，不过在这些内容当中，被大家经常提及的除了区块链的技术体系外，究竟它的核心价值是什么？是更好的性能、更方便的架构、更可靠的技术，还是其他的因素？关于这个问题，在2017年8月14号的区块链开源讨论会上，来自微众、万向、矩阵元的领导总结了一个非常到位的词，那就是“信任”。

很显然，微众银行、万向区块链和矩阵元联合发布共同开发的区块链开源平台BCOS，综合了区块链技术所带来的很多非常好的机制，如各方更好地进行数据协作、身份认证、非对称加密算法、引入技术治理功能、实时的点对点清结算体系，甚至一些去中心化的协作体系。但是从技术的层面讲，中心化的系统其实都可以做到，甚至从纯技术角度来说做得更好。

那么我们该怎么来看BCOS通过开源来把区块链技术的核心问题——信任，通过开源，融入到用户和企业当中去？开源又如何去解决一系列关于“信任”问题：信任是从哪里来的？区块链带来的信任是哪一种信任？这些“信任”在区块链的新生态下还有作用吗？

好，那我们先从三家企业为什么要做一个开源的区块链平台BCOS说起。

三家共同发力 打造基于信任的开源区块链平台BCOS

近两年来，以多方参与、共享资源、智能协同、价值整合、模式透明等为主要特征的“分布式商业”模式开始崛起，为了实现分布式商业的共享与透明规则，以开源为主要特征的分布式技术也得以发挥优势， 区块链技术、分布式账本技术等渐渐成为了前沿科技的核心代表。

矩阵元作为一家由前中国银联老兵孙立林主导的企业，分布式账本技术的领导厂商，在去年通过第一次跟微众银行、万向三方沟通，基于分布式账本技术的清算和结算解决方案获得了微众银行和万向的一致认可，基于技术路线和微众银行的理解上保持着高度一致，推出首个基于信任，开放的区块链开源平台BCOS。

与基于单一信用背书实体的传统信任机制不同，区块链的信任机制是多个参与方对透明和可信规则的共同信任、是对客观信息技术的信任。因此，为了增加信任与充分透明，减少甚至完全剔除人的干预，区块链从诞生开始就是以开源技术的形式出现，大部分的区块链技术平台也皆以开源社区的形式存在。

微众银行副行长兼首席信息官马智涛表示，区块链技术的开源对于整个业界的发展是非常重要的一步，有了这种基础才能把企业的信任机制建立起来，同时也满足了企业的多方面需求。

第一：技术需求

在区块链技术方面，要求对等协作、公平公正、透明可视，这些都是区块链技术本身的特征。有了开源基础，完全把代码公开透明的展示给用户和企业，通过多家参与并定制规则，才能够把对等协作的业务模式运转起来，这也是区块链必须走出的第一步。

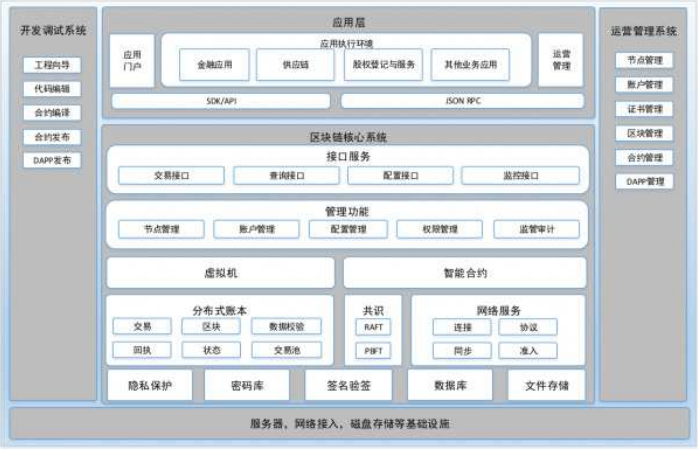
第二：市场需求

对于中国市场，虽然区块链技术在中国市场上炒的非常火热，但是更多的企业没有把精力聚焦在场景应用上，反而集中在区块链技术底层的研究。通过微众、万向、矩阵元三家企业结合力量打造并开放出底层平台，对外开源，让更多的企业把更多的精力放在对区块链场景的应用的落地，是促进国内区块链发展的重要基础。

第三：国情需求

不管是技术还是市场，区块链的发展大部分还是被国外机构所主导，但是中国市场有自己的特色以及诉求是国外机构无法了解和掌握的。通过开源出区块链的底层技术，把国内企业的力量聚焦在一起，打造出一个符合国情的生态圈，一个监管机构和市场都能认可的底层平台，更加助力于我们在整个区块链行业当中站在前沿的地位。

矩阵元创始人孙立林表示，希望通过 BCOS 平台的开源，打造一个符合国情的、低接入门槛的、易于使用的、可规模化商用的企业级区块链开源底层平台，以便于满足各行业应对分布式商业涌现的需求，亦希望通过开源社区的建设，形成一个可持续发展的国际化技术生态圈。

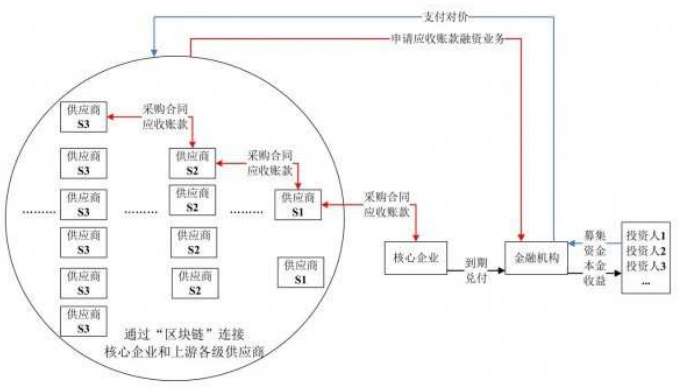


BCOS 平台总体架构图

让“信任”完成第一个供应链金融服务平台

赊账销售这一现象在经济运行中愈加普遍，据国家统计局数据显示，2016 年末我国规模以上工业企业应收账款 12.6 万亿元，各行业账期平均在 90 日左右，这其中蕴含着企业巨大的融资需求。而相比于巨大的应收账款，2015 年我国年商业保理量仅在 2000 亿元左右。可以看出，还有大量融资需求没有被满足，因而供应链金融行业发展空间巨大。

以汽车零部件起家的万向集团旗下的万向区块链董事长兼CEO肖风认为，汽车零部件制造的供应链在传统方式中就已经存在，但是由于供应商资金在供应环节被大量占用，为了解决供应商的态度问题，万向通过协议把供应链重构应有的汽车零部件供应链，采用了BCOS平台的底层技术开发一个供应链金融服务平台，以优惠应收账款融资流程。



以 BCOS 平台底层技术打造的供应链金融服务平台

供应链金融业务非常适合采用区块链与分布式账本技术。首先，业务涉及多个主体，包括核心企业、供应商、金融机构、投资人等，通过区块链将交易信息在多主体间传递；其次，供应链上交易信息通常为商业敏感信息，区块链技术可实现数据加密；最后，通过区块链技术的数据不可篡改性和可追溯性，可有效杜绝供应链融资常见的虚假贸易现象。

与此同时，孙立林对记者透露了矩阵元正联合某区域股权交易中心，基于BCOS平台打造一个服务于中小微企业的新一代股权登记与服务平台，从企业股权登记业务着手，通过新兴技术手段来重构企业信用基础，引导公共服务机构和商业机构共同参与，形成便利中小微企业交易融资的新型生态体系。

安全保障及可运维性，让信任升级

在安全的环节当中，密码学，零知识证明、同态加密等算法的研究扮演了非常重要的角色，很多局部的信任点或信任体中，都需要大量的密码学、同态加密等算法的支持，在区块链的生态当中，密码学和加密算法的研究成为区块链的技术信任的基石之一。

BCOS在在一些强监管或高安全要求的业务场景中，可以引进官方或权威的中央对手方提供信用背书，交易参与方的交易数据明细对中央对手方为全部可见，但对不相关的参与方为不可见，中央对手方负责对交易进行验证并提供面向全联盟链的可信证据。在这种业务场景里，通过物理隔离与通信层的逻辑通道设计，交易明细仅发送给交易牵涉的节点以及中央对手方，从基础层面防止了隐私数据的扩散。

而在区块链的安全问题两件事情是非常重要的，那就是安全与可运维性。对于BCOS来说，肯定也有缺陷，安全永远没有终点，所以通过BCOS的开源，让大家共同检验，逐渐和大家一起去完善。

在可运维性面前，BCOS是一个重视安全、运维商业级的系统，从构建和运行逻辑上都具有较高的一致性，不同节点的软硬件系统基本保持一致。通过标准化的特性给运维人员带来了极大的便利性，使用通用的工具、运维策略和运维流畅度斡对区块链系统进行构建、部署、配置、故障处理，从而降低运维成本以及提升效率。

对于区块链应用的三个技术方向——存储、密码学的隐私保护、共识机制而言。它不应该被一家企业所控制，它本身就是应该开放，通过开放的标准，整个社会都能够参与进来，一起收益，共享成功，完善整个生态，促进各类分布式商业应用场景落地，这才是区块链技术未来最终的路线。 ■

每秒高达千万分发，如何应对直播互动平台中海量消息挑战?

由于直播平台的特点，对系统功能设计的可靠性要求更高，同时，如何在直播火热的当下快速实现直播平台的构建，成为很多企业的现实需求。本文主要分享融云直播互动系统的设计与实践，详细介绍礼物、红包等以消息为基础的互动方案设计思路和实践方案并阐述如何结合融云自身技术优势，助力直播企业快速布局市场。

直播互动平台的特点

直播互动平台的特点是什么呢?在做视频直播时，用户需要跟主播以及其他用户发一些文字消息、点赞消息等等进行互动，这些消息的存在把以前单纯的只是受众于视频的应用视频直播，变成可以进行互动的平台。

用户在平台发消息进行互动，这个互动场景和聊天室类似，就是用户加入聊天室，可收到别人的消息，同时也可发送消息，用户下线时消息不再进行推送，这个场景实际上可以满足简单的直播互动。

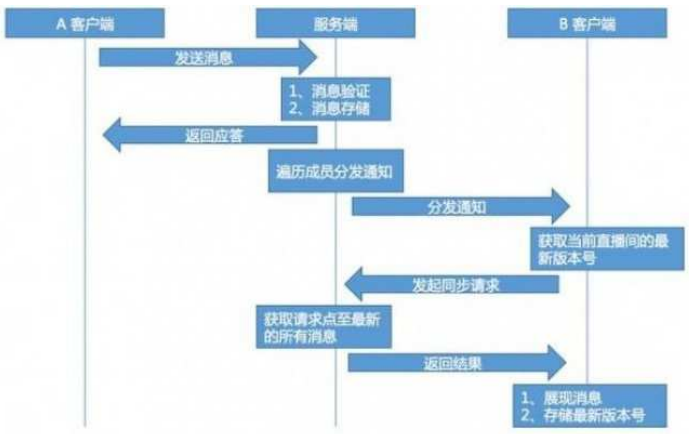
相较其他 IM 来讲，没有离线消息存储，也没有离线消息 Push，在业务形态来讲，聊天室比较简单一些。

如下是直播互动平台的主要特性:

- 直播互动平台的用户量实际是无上限的，因为不可能去限制一个主播加入观众的成员数量。
- 通过把消息进行分级控制的方式来保证礼物、红包等消息传输的可靠性。
- 瞬时消息吞吐量大，融云某个聊天室曾经一秒钟产生过近千万的消息。
- 针对伸缩及稳定性方面的高要求。

消息平台的逻辑设计及设计原理

消息是直播互动系统最重要的元素，消息处理做的好不好会直接影响用户的体验。融云的消息平台逻辑设计是怎样的呢?



如上图，我用一张时序图针对这个问题进行了解答。A 客户端是发消息方，服务端是融云整个后端平台，B 客户端是接收消息方。

首先，客户端向服务端发送消息，服务端接收消息先进行消息验证，消息验证里面包含了高危词、敏感词的过滤，反垃圾系统对消息进行反垃

圾的过滤及权限验证。

当消息通过之后，对消息优先进行存储，之后会给 A 客户端返回应答，告知消息已经发送成功，这时服务端会遍历成员分发通知。

值得注意的是，现在还没有向客户端发送消息实体，当接收客户端接收消息通知时，会从自己本地存储里面拿到当前直播间里面最新的消息版本号。

同时向服务器发起同步请求，服务器获取客户提交的版本号之后，会以这个版本号为起点，到最新版本号把所有消息全部拿回来返回给客户端，客户端收到这个消息进行消息展示以及存储这个最新版本号。

整个通信流程，版本号获取方式，跟常用版本控制软件比较类似。之所以把流程变得这么复杂，在发送完通知之后再同步消息，主要原因如下：

- 通知的推送重试可以依赖 TCP 的重试，无须逻辑保障。
- 服务器端存储的消息顺序与到达的客户端顺序一致。
- 当消息密集时可以将多条消息进行合并。
- 可以控制下发的消息条数。
- 以存储分级实现消息分级。

不以消息数量衡量服务器规模，以在线用户数量做为考量。

消息平台在服务端实现细节及最佳应用实践

直播消息系统在服务端的实现细节，主要从数据预处理、消息分级、消息存储和消息分发队列四方面进行介绍，但在实际服务开发过程中，还有很多优化。

数据预处理

当服务端接收消息之后，优先对数据进行序列化以及存储，序列化消息和要下发的客户端数据要保持一致，通过这种预处理的方式，CPU 的使用率在当前基础上降低了 30%。

消息分级存储

消息类型按照中、高、低序列进行分级存储

Skiplist 转化为环形队列存储

每条消息有唯一且递增版本号，消息按照版本号顺序存储。这样的数据在内存存储情况下，跳表是一种非常合适的数据结构。

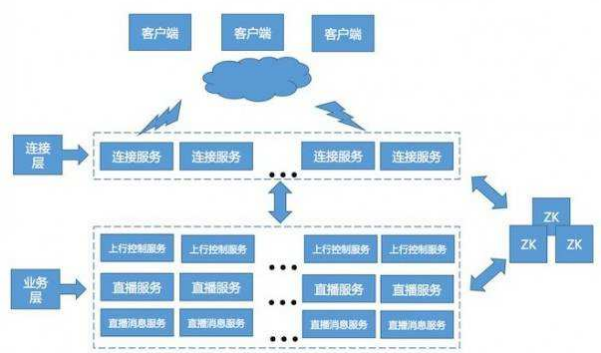
但是，跳表时间复杂度和空间复杂度比较大，同时往跳表中插入数据时使用的锁范围非常大，会对服务器产生一些额外的开销。之后，由跳表演进成一种类似环形队列的数据结构，会对服务器性能开销提升很多。

Map与队列构建复合数据结构 降低无意义消费

服务端下发通知，对于通知来讲在这一时刻无论有多少条消息，每个用户只有一条通知，通过 Map 与队列构建这样的复合结构进行存储。

队列里面存储是数据的指针，Map 存储用于下发的通知实体，Map 使用的 key 为用户标识，这样可以降低无意义的消费。

融云线上直播互动平台2.1整体架构



如上图，是融云线上 2.1 架构图，可分为连接层、业务层和存储层等，连接层和业务层构建成整个大的集群，然后进行服务。

连接层主要是负责客户端与服务器保持长连接，同时将客户端的协议与内部服务的协议进行相互转化。

业务层负责 IM 相关业务处理，采用的是微服务架构，线上服务有 40 多个类型，但对于直播互动平台包含如下三项服务：

上行控制服务。主要目的是用于处理接收的上行客户端的消息，进行敏感词和高危词的过滤。这个服务的负载方式采用的是随机分配。

直播服务。负责维护直播间的成员关系和负责接收上行控制服务给它的消息，上行控制服务会先期进行消息抛弃，抛弃到直播服务可以接收的范围之内，然后将消息下发到直播服务，直播服务再广播到直播消息服务。

直播消息服务。负责给用户进行分发通知，是以用户 ID 进行一致的哈希方式进行负载。每个服务节点包含部分用户关系。

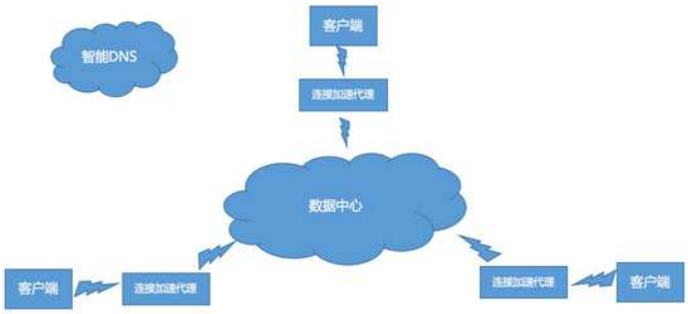
整个直播消息服务构建了一个整体成员关系，等同于直播消息服务上的一个节点。这层除分发通知外，还负责客户端同步拉取获取的消息。

存储层主要职责是用来存储各种消息，数据等，含多个数据库。

线上 2.1 架构和之前的架构相比，降低了直播服务压力和上行总量的控制，但面临的问题是需要更高的网络质量要求。

直播互动平台的网络优化

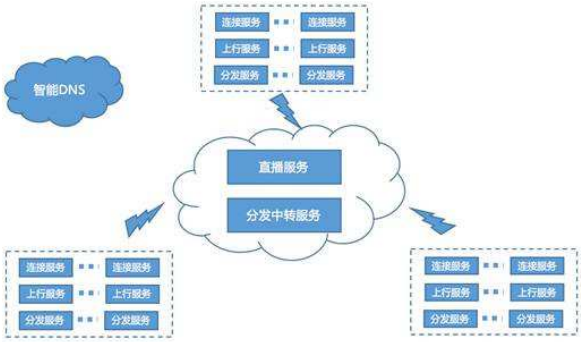
如下图，是链路架构 1.0：



链路架构 1.0，线上进行网络加速主要是加速海外业务。融云选购了第三方海外专线链路，涉及全球五个国家，覆盖欧洲、美洲以及中东、日本等等。

实现国际链路优化，就近区域接入的同时，又出现专线的流量过大，成本过高，跨国消息延时等问题。

如下图，是链路架构 2.0：



链路架构 2.0 采用模拟 CDN 模式进行数据分发，专线内只跑上行消息和广播消息的方式。

■

来源：51CTO
译者：刘妮娜

七个聊天机器人构建平台：轻松打造自己的机器人

当今大家最为熟悉的聊天机器人就是苹果手机里的助手Siri。那么，如何打造一个自己的聊天机器人呢？你需要一个构建平台。

聊天机器人（chatterbot）是用来模拟人类对话或聊天的程序。世界上最早的聊天机器人诞生于20世纪80年代，名为”阿尔贝特”，用BASIC语言编写而成。今天的互联网上，已出现”比利”、”艾丽斯”等聊天机器人。当今，大家最为熟悉的聊天机器人就是苹果手机里的助手Siri。那么，如何打造一个自己的聊天机器人呢？你需要一个构建平台。今天就请大家跟随笔者一起看看这七个强大的聊天机器人构建平台。

目前一些主流的聊天机器人发布平台包括FBMessenger、Kik、Slack、Telegram和WeChat。聊天机器人发布平台是打造聊天机器人的媒介、工具或应用程序也通过它可以创建聊天机器人。我们有像Chatfuel、Botsify这样的聊天机器人开发平台（下文中我们会对其进行了解）。聊天平台能够帮助你为机器人添加更多功能，提供机械学习能力和API集成等等。

1.Chat fuel

使用Chatfuel这个平台，即使是没有任何代码基础的素人也可以轻松掌握。你可以在Facebookmessager使用Chatfuel打造自己的聊天机器人。它可以提供一些功能，比如添加内容卡片和与你的追随者自动分享，在表单的Messenger聊天中收集信息，让用户请求信息并与按钮交互，而且完全免费。

2.Bot sify

Botsify是另一种受欢迎的FacebookMessenger聊天平台。它使用一个拖放模版创建机器人。当你想要创建机器人时，可以支持通过插件的简单集成、人工智能、机器学习和集成分析等功能。虽然机器学习和集成人工智能对于没有技术知识的人群会有些困难，但是这个平台所提供的拖拽功能让更多人可以自己制

造一个聊天机器人。Botsify平台可以免费制造一个机器人，第二个开始需要收费。

3.Flow XO

FlowXo是一个好评率很高的聊天平台，它拥有易于使用的可视化编辑器。使用FlowXO可以支持很多平台，而不用反复去制作。但它的弊端在于仅限于一定数量的会话，在该会话中必须创建订阅以供进一步使用。

4. Beep Boop

BeepBoop平台提供端到端的开发经验，为机器人构建者提供了宝贵的参考依据。在BeepBoop平台，你可以构建出伟大的机器人。这个平台是运行Slack机器人的最佳平台。

5.Bottr

Botter是在一个有趣的概念，可以帮助你在Facebook、Twitter、LinkedIn上构建有趣的聊天机器人。而且它非常的简单和迅速，不到5分钟的时间里，你就可以在Botter上构建一个机器人。它会根据你的Facebook数据自动设置问题。Bottr支持在你的网站上嵌入你的机器人。你也可以从一个介质添加数据，WordPress或维基百科，实现更好的数据覆盖。

6.Motion ai

Motionai是一个直观的聊天平台，可以在FBMessager、Slack、Smooch和你的网站上建立、训练和使用机器人。它提供您创建会话的流程图，以便直观地了解机器人查询的结果。还可以链接机器人到Slack、FacebookMessager、Go这样的信息服务平台。motion.ai允许您从几种集成接口直接部署Node.js代码。

7.chatty people

chattypeople平台上有预设好的聊天机器人模型。这些模型已经对客户服务、食品企业类的电子商务做好准备。如果你选择电子商务的聊天机器人，那么你只需要添加产品，问答，和一些常规设置，就可以完成了。

作者：MitulMakadia

原文链接：https://dzone.com/articles/which-are-the-7-powerful-platforms-to-build-a-chat

■

不看后悔：提升网站转化率的Web设计忠告

来源：51CTO
译者：刘妮娜

好的UI能够反映出网站的高水准以及带来良好收益。从设计师的角度看来，了解网站的受众群体非常重要，这决定着你的设计风格。

电子商务对实体商业的促进作用毋庸置疑，然而就网站本身而言，UI设计在很大程度上将决定你能从营销中获得什么效果。好的UI能够反映出网站的高水准以及带来良好收益。从设计师的角度看来，了解网站的受众群体非常重要，这决定着你的设计风格。你的网站有多大吸引力，可以吸引多少访问者，他们之中有多少人会最终在你的网站购买商品，这是需要考虑的问题。

下面是一些网页设计技巧，可以帮助你提高转化率：

1.让用户体验到优先权

如果你的网站用户体验不佳，那么还没到考虑转化率的时候。许多设计师只注重设计风格的别出心裁，他们想要做出令人惊叹的页面设计，但却忽略了访问者、使用者的实际需求。你必须了解使用者是谁，让他们愿意与你互动。确定他们是如何到达你的网站，以及他们在寻找什么。一旦你掌握了所有这些信息，想想他们能从你的站点中得到什么，并努力向他们提供这些信息。提升用户体验需要在很多方面下功夫，包括用户的需求、他们的喜好以及在网站上的行为。

2.让你的网站更易读

当网上购物者在网上寻找业务、产品或服务时，他们主要是寻找与他们感兴趣的产品或服务，以及相关信息。有些人在不同的网站上寻找不同的选择，比较后选择最好的。有些购物者只

是简单地浏览网上信息，以便收集他们想要购买的产品信息。这样，他们就可以做出明智的选择。现在大多数买家都在网上搜索他们想买的东西，生意人就需要投其所好，提供他们想要的东西，以易读的方式设计网站。你要想法设法让这些购物者在你的网站上停留更长的时间，他们才可能最终从你那里购买。所以，不要让你的网站内容太过拥挤，要让读者能快速的浏览信息。

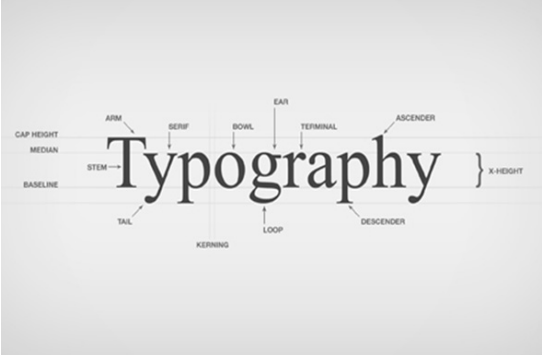
3.选择适合的颜色

色彩在网页设计中非常重要。它是设计要素之一，在很大程度上影响转换率。许多在线购物者根据颜色来判断一个网站。颜色被用来决定网站的主题和调性。每种颜色都有不同的意义，所以必须谨慎设计网页的颜色。有些颜色会让你的访问者一来到页面就点击后退按钮。有些颜色则会让他们兴奋，但仍然可能对你的内容失去兴趣。所以你必须深入的了解你的网站和受众群体，想想哪些颜色会使他们兴奋，能引导他们继续阅读。提前为你的网站设置主题和调性，并选择一种颜色与之匹配。让颜色成为你的工具、你的武器，俘获网友的心，提升转化率吧！

4.让你的网站反应灵敏

今天，大家都使用移动设备访问网络，如果你在设计网站的时候不考虑各种设备和更小的屏幕，你将会失掉市场。对于许多网络用户来说，一个无响应的网站是一个巨大的坑，尤其是那些使用移动设备的用户。很明显，当他们试图在较小的小工具上查看某些页面时，很难导航到网站。移动Web浏览是现在的趋势，许多消费者都在网络上，他们的移动设备试图购买、查询和下载应用程序。您需要一个响应性的设计，因为这将确保您的网站能够适应任何类型的屏幕。这样，潜在的买家可以清楚地看到你在卖什么，而不必在一个很小的屏幕上纠结。

5.选择正确的版式



有一种因素经常被商业人士忽视，但排版在提高网站转换率方面起着重要作用：那就是字体。如果你不想受到来自你网站的任何人的批评，请避免使用奇怪的字体。另一方面，那些看起来很偏门的字体不会吸引任何读者。有这么多大气恢弘的字体，你尽可以在你的网站上使用。那些看起来充满权威性的字体可以保持页面的可读性。这些是你应该在你的网站上使用的。通过一个合适的字体，完美诠释你的网站。谷歌有一个全面的Web字体类型库，它可以指导你找到适合你网站的正确字体。

拥有一个漂亮的网站还不够，你必须确保它有一定的转化率，以帮助你的企业最终实现盈利目标。希望我们在上文中谈到的内容能帮助你的网站提升转化率，实现更好的营收。

原文链接：http://codecondo.com/web-design-tips-to-help-you-improve-your-conversion-rate/

■

AWS披露面向Amazon S3的AI监控方案

AWS 近日正式上线Amazon Macie——这项服务旨在利用机器学习机制帮助客户预防敏感数据意外泄露以及未经授权的Amazon S3存储数据访问。该公司表示，Amazon Macie将在今年晚些时候支持更多其它AWS存储服务选项。

在AWS的S3平台当中，Amazon Macie负责利用自然语言处理机制发现并分类敏感数据、识别个人身份信息以及私钥与信用卡信息等因素。Macie服务还将不断监控异常的数据访问活动。AWS人工智能总经理Matt Wood表示，此类异常活动将面向客户安全团队触发相应警报。

Macie将帮助用户了解与数据相关的其它风险，包括在客户无意对敏感数据进行外部访问或者非安全方式存储时，提供自动警报提醒。除了分析数据以了解用户身份、访问位置与访问次数等历史模式，以及识别及分类敏感数据之外，Amazon Macie还提供一套专门用于追踪用户活动及监控数据安全性的仪表板。

Macie控制台还允许定义自动化修复操作，例如重置访问控制列表或者重置密码内容。同时,AWS方面公布了一系列与其它服务相关的改进：

1. 正式启动AWS Migration Hub，旨在追踪来自数据中心并指向AWS的应用程序迁移活动。并提供一整套AWS及合作伙伴提供迁移工具。

2. 在Amazon弹性文件系统当中对AWS静态数据进行加密，意味着用户能够选择一套密钥以加密文件内容。此功能目前已经在各支持弹性文件系统的服务区中上线。

3. 对CloudHSM(即硬件安全模块)进行完全重写，旨在满足敏感数据提出的安全性与合规性要求。CloudHSM目前成为一项可扩展的受控服务，且内置有配置、补丁管理、高可用性以及备份等功能特性。此项服务目前亦以按需计费方式交付，意味着无需任何前期投入。另外，CloudHSM支持FIPS 142-2 Level 3，同时包含多项对HSM物理访问或修改尝试进行检测及响应的机制。

4. AWS Config服务当中迎来多项新规则，用于评估AWS配置方案。各项新规则有助于保护S3存储桶，其中包括识别各允许全局读取及/或写入的存储桶。

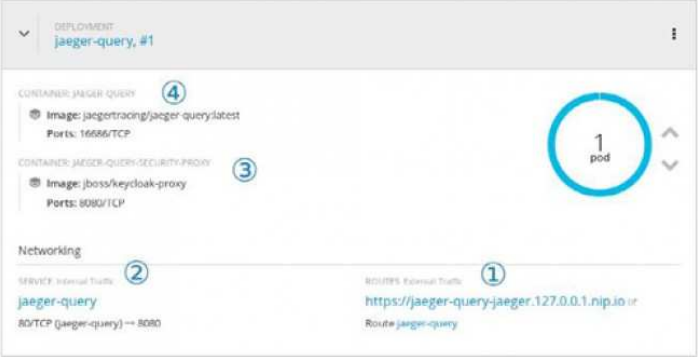
5. AWS CloudTrail(一项面向各AWS帐户的治理、合规与审计服务)现在默认为全部客户启用，且能够提供过去七天内的帐户活动明细。用户不再需要为此项服务作出任何配置。

6. AWS Glue(Amazon云中的一项提取、转换与加载服务)目前已经全面上线。这项“无服务器”服务能够帮助用户摆脱资源供应及管理方面的相关负担；如今大家只需要承担Glue运行带来的资源成本即可。 ■

来源：51CTO
译者：陈峻

用Sidecar安全代理保护Jaeger用户界面

在Jaeger的产品部署中，我们限制访问Jaeger的查询服务，包括用户界面(UI)，是非常有益的。例如，您可能有各种内部安全的要求，只允许某些群组能够访问跟踪数据，或者您可能将Jaeger部署到了公有云之上。在真正的微服务(microservices)方式里，一种可能的方法是给Jaeger查询服务添加一个Sidecar作为安全代理。各种入向请求都被传递到我们的Sidecar，而非直接到达Jaeger的查询服务上。而Sidecar将负责执行各种身份验证和授权的限制。



如上图所示：传入的HTTP请求到达路径(route)①，它使用内部服务②来进行解析，并与安全代理③进行通信。一旦请求通过验证，且所有安全的限制得到了满足，则该请求就到达Jaeger④。

出于演示的目的，我们将使用Keycloak(译者注：Keycloak是一个为浏览器和RESTful Web服务提供SSO的集成)作为自己的安全解决方案，当然其他任何安全代理也都适用于这个理念。当然如果您并不修改Red Hat的SSO(Single Sign-On)，这个演示也能够运作。因此对于本次操作练习，我们需要如下：

- 一台Keycloak(或是Red Hat的SSO)服务器的运行实例。我们定义它的位置为$ { REDHAT_SSO_URL }
- 一个OpenShift的集群，我们将用来运行Jaeger的后端组件。它就像oc cluster up一样容易
- 一份Jaeger OpenShift产品模板的本地克隆

注意：我们并不会试图对组件之间(比如说从代理到收集器)的通信进行安全加固。而对于此类场景，我们完全可以使用其他技术来实现，例如相互间使用证书来进行认证，采用istio(译者注：istio是一个开源项目，提供统一的连接、安全、管理和监控微服务)，或其他类似的工具。

准备Keycloak

对于该演示，我们将直接在主机上通过Docker来运行Keycloak。这是为了强调Keycloak并不需要像我们的Jaeger后端那样运行在同一个OpenShift集群之上。

下面的命令用来在主机上启动一个合适的Keycloak服务器。如果你已经有了自己的

Keycloak或Red Hat的SSO服务器的话，则完全跳过这一步骤。

```
1.  docker run --rm --name keycloak-server -e KEYCLOAK_USER=admin -e KEYCLOAK_PASSWORD=password -p 8080:8080 jboss/keycloak
```

一旦Keycloak服务器启动并运行，我们就可以创建一个Jaeger的域(realm)了:

1. 请使用用户名：admin和密码：password，登录到Keycloak(http://<YOUR_IP>:8080/auth/admin/master/console)。

2. 在左上角Select realm中用鼠标点击Add realm。将其命名为jaeger，然后点击Create。

3. 在Clients上，点击Create，将proxy-jaeger设置为名称并保存它。

4. 设置Access Type为confidential，并用*代表Valid Redirect URIs，并保存之。您可能需要在生产环境中进行微调，否则可能会暴露出已知的“未验证的重定向和转发”(Unvalidated Redirects and Forwards，译者注：源自OWASP TOP10 2010的A10)攻击。

5. 打开Installation选项卡并选择Keycloak OIDC JSON，然后复制其显示的JSON。如下所示，不过auth-server-url和secret的值会有所不同。

```
1.  {
2.    “realm”: “jaeger”,
3.    “auth-server-url”: “http://192.168.2.111:8080/auth”,
4.    “ssl-required”: “external”,
5.    “resource”: “proxy-jaeger”,
6.    “credentials”: {
7.      “secret”: “7f201319-1dfd-43cc-9838-057dac439046”
8.    }
9.  }
```

最后，让我们创建一个角色和用户，使得我们能够登录Jaeger的查询服务:

1. 在左下侧菜单-Configure的下面，请打开Roles页面，然后单击Add role。

2. 将角色的名称设置为user，并点击Save。

3. 在左下侧菜单-Manage的下面，请打开Users页面，然后单击Add user。

4. 按照您的设想去填写该表格，并将Email verified设置为ON，在并点击Save。

5. 为该用户打开Credentials选项卡，并设置一个(临时的或非临时的)密码。

6. 打开该用户的Role mappings选项卡，从Available Roles列表中选择角色为user，再点击Add。

准备OpenShift

对于该演示，我们假设您已经有一个OpenShift集群正在运行了。如果还没有的话，那么您可能需要参考minishift之类的工具了(译者注：minishift是一个通过虚拟机来模拟OpenShift集群的工具)。如果您正在运行最新版本的Fedora、CentOS或Red Hat Enterprise Linux，您可能需要安装包origin-clients，并运行oc cluster up --version=latest。这样您就能有一个基本的、运行在本地的OpenShift集群。

为了方便演示，我们将添加cluster-admin的权限给developer用户。同时我们创建如下的Jaeger命名空间:

```
1.  oc login -u system:admin
2.  oc new-project jaeger
```

```
3.   oc adm policy add-cluster-role-to-user cluster-admin developer -n jaeger
4.   oc login -u developer
```

准备Jaeger OpenShift模板

我们将使用Jaeger OpenShift产品模板(https://github.com/jaegertracing/jaeger-openshift/blob/master/production/jaeger-production-template.yml)作为开始:克隆整个存储库，或者得到本地版本的模板。

第一步是添加sidecar容器到query-deployment对象里。在containers列表下，我们指定了jaeger-query之后，就可以添加如下的sidecar代码:

```
1.   - image: jboss/keycloak-proxy
2.             name: ${JAEGER_SERVICE_NAME}-query-security-proxy
3.             volumeMounts:
4.             - mountPath: /opt/jboss/conf
5.               name: security-proxy-configuration-volume
6.             ports:
7.             - containerPort: 8080
8.               protocol: TCP
9.             readinessProbe:
10.              httpGet:
11.                path: “/”
12.                port: 8080
```

注意：该容器将volumeMount的名称指定为security-proxy-configuration-volume，我们将使用它来存储代理的配置文件。您可以在spec/template/spec节点下为query-deployment指定容量，并同样设定dnsPolicy的属性(它应该是在前续的代码片段中):

```
1.   volumes:
2.             - configMap:
3.                 name: ${JAEGER_SERVICE_NAME}-configuration
4.                 items:
5.                   - key: proxy
6.                     path: proxy.json
7.               name: security-proxy-configuration-volume
```

现在，我们需要来指定具有代理配置条目的ConfigMap。要做到这一点，我们应当给该模板添加一个新的顶层项目。在这里，我们建议您将其放置得越接近于其被使用之处越好。例如，就放在query-deployment的前面:

```
1.   - apiVersion: v1
2.     kind: ConfigMap
3.     metadata:
4.       name: ${JAEGER_SERVICE_NAME}-configuration
5.       labels:
6.         app: jaeger
7.         jaeger-infra: security-proxy-configuration
8.     data:
9.       proxy: |
10.        {
11.            “target-url”: “http://localhost:16686”,
12.            “bind-address”: “0.0.0.0”,
13.            “http-port”: “8080”,
14.            “applications”: [
15.                {
16.                    “base-path”: “/”,
17.                    “adapter-config”: {
18.                      “realm”: “jaeger”,
19.                      “auth-server-url”:
```

```
"${REDHAT_SSO_URL}",
20.                     "ssl-required": "external",
21.                     "resource": "proxy-jaeger",
22.                     "credentials": {
23.                       "secret": "THE-SECRET-FROM-INSTALLATION-FILE"
24.                     }
25.                   }
26.                 ,
27.                 "constraints": [
28.                     {
29.                         "pattern": "/*",
30.                         "roles-allowed": [
31.                             "user"
32.                         ]
33.                     }
34.                   ]
35.                 }
36.             ]
37.         }
```

请注意：我们只允许具有user角色的用户登录我们的Jaeger用户界面。在真实的场景中，您可能想要调整之，以适应您自己的设置。例如，您的用户数据可能来自LDAP，而您只想允许来自特定LDAP组的用户去访问Jaeger的用户界面。

各个credentials中的secret应当与我们从本练习最开始的Keycloak中所获得的secret相匹配。颇具好奇心的读者您也许会注意到，我们在auth-server-url属性下所提到了模板参数REDHAT_SSO_URL。改变您的Keycloak服务器，或是我们指定一个模板参数，都能允许我们在部署时对它进行设置。在该模板的parameters部分，我们可以添加以下属性：

- description: The URL to the Red Hat SSO / Keycloak server

displayName: Red Hat SSO URL

name: REDHAT_SSO_URL

required: true

value: http://THE-URL-FROM-THE-INSTALLATION-FILE:8080/auth

可见这个值应该是一个您的浏览器和sidecar都能访问到的位置，就像您主机的局域网IP地址(192.x.10.x)一样。显然，使用Localhost/127.x是无法工作的。

作为最后一步，我们需要更改服务引导请求到端口8080(即为代理)上，而不是16686。你可以更改服务名为query-service的属性targetPort，将其设置到8080：

```
1.   - apiVersion: v1
2.     kind: Service
3.     metadata:
4.       name: ${JAEGER_SERVICE_NAME}-query
5.       labels:
6.         app: jaeger
7.         jaeger-infra: query-service
8.     spec:
9.       ports:
10.      - name: jaeger-query
11.        port: 80
12.        protocol: TCP
13.        targetPort: 8080
14.      selector:
15.        jaeger-infra: query-pod
```

```
16.      type: LoadBalancer
```

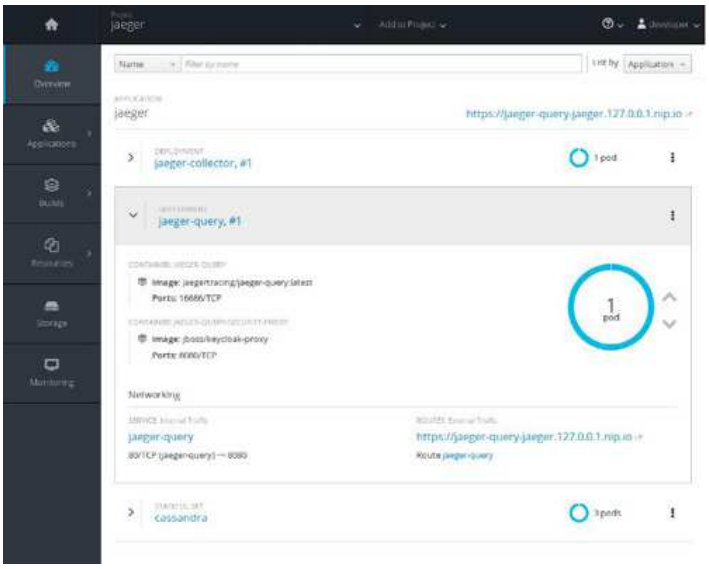
作为参考，您可以在博客链接--https://github.com/jaegertracing/jaeger-openshift/blob/KeycloakSecuringUI/production/jaeger-production-template.yml里看到完整的模板文件。

部署

现在我们已经一切准备就绪了，那么就开始部署Jaeger到我们的OpenShift集群上吧。请您在前续步骤里存储了YAML文件的相同目录中，运行以下的命令。我们在此引用到的名称是jaeger-production-template.yml:

```
1.   oc process -f jaeger-production-template.
yml | oc create -n jaeger -f -
```

在开始运行的头几分钟内，如果pod的jaeger-query和jaeger-collector出现失败的话并无大碍，Cassandra(译者注：Cassandra是一套开源分布式数据库管理系统，用于储存特别大的数据)仍将会完成启动。最终，该服务也会启动并运行，如下图所示。



一旦准备好了接收服务，请点击路径(route)的URL(https://jaeger-query-jaeger.127.0.0.1.nip.io)。一个由Keycloak服务器所提供的登录界面就会呈现在您眼前。请使用您在前续步骤中设置好的凭证进行登录，之后您就可以顺利到达Jaeger的用户界面了。

结论

在这个练习中，我们了解到了如何为Jaeger的查询pod添加一个sidecar作为安全代理。它让所有的入向请求都传递到该sidecar，而Keycloak的所有特性，例如：双因素认证、服务帐户、单点登录、暴力攻击保护、LDAP支持、等都仍然是透明可用的。

■

挨踢部落故事汇(22)

从逃离北上广到逃回北上广

司马雪是玖哥的学长，先后在国内两家top 10大厂担任资深Android工程师，对Android技术有如痴如醉的热情，并且乐于帮助同行，最难得的是他还有一个漂亮的女朋友，不愧是Android程序员中的人生赢家。玖哥将他的经历稍作加工后用明清小说的笔法写出来，希望每个读到这篇文章的Android工程师都能走他的路。

第一回：小庙无地容巨擘 大厂有礼迎硕士

诗曰：

老板抠门巧计乖，却将忠义苦挤排。

基础扎实绩点高，苍天岂能误人才。

司马雪2008年考入合肥工业大学，2012年保研本校，2014年进入合肥某小公司实习。

司马雪一开始选择的是JavaEE方向，他在实习期间就以专家的标准严格要求自己，不但让服务端的内核稳定度提升了好几个档次，还让内存消耗下降了好多个数量级。完成本职工作后还帮助运维那边修复了几次硬件的故障。不料老板嫉贤妒能成性，只爱奴才，不要人才。这个老板不但不给司马雪升职加薪，还侮辱司马雪：““雪”是“宕”的繁体字，你越俎代庖搞运维不怕把服务器搞宕机？”信而见疑、怀才不遇的司马

雳愤而辞职。司马雳辞职刚好赶上某大厂（国内top 10）招Android实习生，他突击学习了几天Android就去面试了。

一面的时候主要考察Android和Java的基础知识，比如Java的数据类型、运算符优先级和Android的布局以及生命周期，基础扎实的司马雳都能准确无误地回答。

二面的时候面试官问了许多数据结构和算法、设计模式、架构的方面的问题，司马雳不但能画出好几种设计模式的UML图，还对MVC、MVP、MVVM的区别和优缺点发表了自己的看法。

最后HR面的时候收了司马雳的成绩单复印件，对他4.0的GPA和专业top 5%感到很满意。

司马雳通过了面试，顺利成为该厂的Android实习生。不久，司马雳以专业top 5%的优异成绩硕士毕业，转正成为Android工程师。而那家小公司呢，不知道什么时候就因为bug太多导致用户严重流失而倒闭了。

异史氏曰：Java玩得6的学生选择Android作为发展方向是一个明智的抉择，这个世界正处于、并将长期处于移动互联网时代。校招面试除了考察Android和Java基础知识，还考察你学习的基础理论课程的知识，最后通常还要收成绩单的复印件。大厂的校招面试一般有固定的问题和模式，比如阿里校招笔试题来自《技术之瞳》，微软校招笔试题来自《编程之美》等，尽管临阵磨枪也有一定的成功几率，但别忘了玩3年LOL的枪不可能比刷6年LeetCode的枪更快。（时间没错，沉迷LOL的大学生都考不上硕士）

第二回：荣升高级愁田舍 游子低头思莼鲈

诗曰：

当年许汜初炒房，羞无才气见刘郎。

如今司马有远志，却愁无房迎新娘。

炒房团自古有之，祖师爷是东汉末年的许汜。当年许汜空有国士之名，却全无救世之意，整天就知道求田问舍，因此被胸襟海阔，志向山高，忧国忘家的刘备鄙视。此事有辛弃疾《水龙吟》为证：“求田问舍，怕应羞见，刘郎才气。”

司马雳和许汜的志向有天壤之别，他的梦想就是把有限的生命投入无限的Android技术中，他不甘心每天只做UI的微调，他发现App存在很大的改进空间，并付诸行动。

司马雳对App的代码进行了大规模重构，在MVP与MVVM中选择了MVP架构，并自主研发了一套的网络请求框架（结合了OKHttp和Gson，可以理解为国产的Retrofit）（该框架不开源，类似框架的源码：https://github.com/qiujuer/OkHttpPacker）代替OKHttp。这样一来不但减少了大量冗余代码，层次结构也变得更加清晰。

订单状态这种需要和服务器实时同步的数据，以前一直用每秒一次的轮询，司马雳发现这是App又费电又费流量的祸根，采取了用推送代替轮询的解决方案。

APK瘦身也是Android性能优化的重要组成部分，司马雳去掉了很多不必要依赖和重复的工具类，让APK打包后的体量轻了一半。

两年后，这个App完成了从3.3到6.3共13个版本的迭代，App的启动速度提升了120%，Crash率也由8‰降低到1‰。立下汗马功劳的司马雳被任命为项目组长、技术指导、高级工程师。

尽管司马雳工作兢兢业业，也为我国的开源事业添砖加瓦，还写技术博客帮助了很多人。北京

高昂的房价却让他有点羡慕那个被他鄙视了多年的许汜。

异史氏曰：Android工程师（在没有“高级”等前缀时）每天最多的工作都是UI的改来改去，想在平凡的工作中取得不平庸的业绩就要付出努力，提升App的性能是有效途径之一。贡献开源代码和分享技术文章的时间也是可以挤出来的。如果在工作中有出色的成绩，升职加薪只是时间问题。

第三回：桑梓惜别因缘浅 楷模入职即资深

诗曰：

北京买房要筑台，还有堵车与雾霾。

回到省会想定居，一问工资又回来。

北京的房价、堵车、雾霾逼得司马雪决定裸辞，逃离北上广，他的目的地是上了七年学的合肥。他刚辞职，老东家的最大竞争对手（全国top 20）和另一家全国top 10的大厂都邀请他去面试。他两个月来参加了这两家大厂和合肥当地两家大厂的面试。

面试官首先询问的总是Java和Android的高级特性，Java的高级特性主要有JVM模型、类加载机制和GC原理等，Android的高级特性主要有几大FLAG和LauchMode的区别和使用场景、Binder的引用和实体以及权限系统的交互等。司马雪对技术钻研很有深度，总是对答如流。

面试的第二阶段就是让司马雪自己去讲他做过的项目，然后面试官会冷不丁的让他去解释其中某一部分，有时候让他解释当时为什么要这么做，有时候问他现在觉得有没有更好的办法。司马雪处理问题的思路和解决问题的能力给面试官留下了深刻印象。

面试官有时候会问一些该企业所在行业需要关注的Android技术，比如研究输入法的公司会询问他Android手势和多点触摸，研究物联网的公司会问他Bluetooth相关知识等。因为司马雪广泛涉猎Android知识，他的技术广度也让面试官啧啧赞叹。

每次面试官问司马雪：“你还有什么想和我说的吗？”的时候，司马雪就把他的技术博客和开源项目一股脑砸向面试官，面试官是司马雪的粉丝或者面试官正在用司马雪的开源项目的情况发生了好几次。

司马雪在合肥拿到了两个资深Android工程师的offer，尽管合肥房价不到北京的1/3，可是最多30W的年薪让司马雪在合肥买房遥遥无期。这时司马雪接到了一个电话，逃离北京前面试的全国top 10的大厂给了他60W年薪和项目经理、技术经理、资深工程师的title。他决定逃回北上广，逃离北上广的计划刚开始就结束了。

异史氏曰：有多年工作经验的求职者几乎不需要在求职网站上投简历，想让你做他的同时的同行朋友会内推你。社招面试和校招面试是不同的，社招面试没有固定的问题和模式，临时抱佛脚是行不通的。社招面试基本都会考察这几个问题：第一个阶段是Android和Java的高级特性，考察技术深度；第二个阶段是讲述自己的项目，并在中间穿插着问题，考察解决问题的经验；第三个阶段（未必有）是问该公司所在行业需要掌握的Android知识，考察技术广度和快速上手情况。技术博客和开源项目是很重要的加分项，如果平时不积累、不分享，求职者会失去很多机会。二三线城市的房价更亲民，但工资非常不人性，逃离北上广需谨慎。

第四回：专家立功施小计 淑女出闺成大礼

诗曰：

经验丰富技术精，消除隐患立大功。

全球大会登讲台，抱得淑女入后宫。

话说司马雪所在的团队负责公司的Android客户端的安全工作，工作内容包含保活、防拦截、防篡改和防反编译等工作。他发现自家App存在不少可能被恶意利用的隐患：

首先，坏人可以通过NotificationListenerService拦截自家App的推送，给用户造成自家App没有推送的假象。

其次，在上一条基础上。坏人可以在虚拟机里运行一个窃取推送App，收到自家App推送后，用AccessibilityService打开Notification对应的Activity，找到里面的WebView，然后取得链接及网页中的内容，稍加修改（换logo、改名字）推送给坏人自己的App。

此外公司的微信公众号打开自己App需要用浏览器打开链接，这也给坏人以可乘之机。

更有甚者，坏人最丧心病狂的手段就是卸载了自家App后，在肉鸡中安装一个与自家App的packgaeName一致，但Signature不一样而且没有launcher的假App，这样肉鸡的用户永远也安装不了自家App，而且还不能用长按拖拽桌面icon进垃圾桶的方式删除假App。

司马雪花了两年时间为公司消除了以上隐患，荣升架构师、技术专家（时间没错，这是小说，出现未来时间很正常）。各大IT论坛、IT活动的聘书和邀请函也如雪片般飞来，不是邀请他做特约作者，就是邀请他当讲座嘉宾。

某IT大会在北京举行，司马雪作为特约嘉宾在台上妙语连珠、口若悬河。大会期间，司马雪结识了某IT社区的技术编辑椎名，这位淑女负责大会的后勤和报道工作。椎名不但知书达理、兰心蕙质，相貌也倾国倾城，有诗为证：

乐天浔阳歌《长恨》，陈王洛水赋《感甄》。

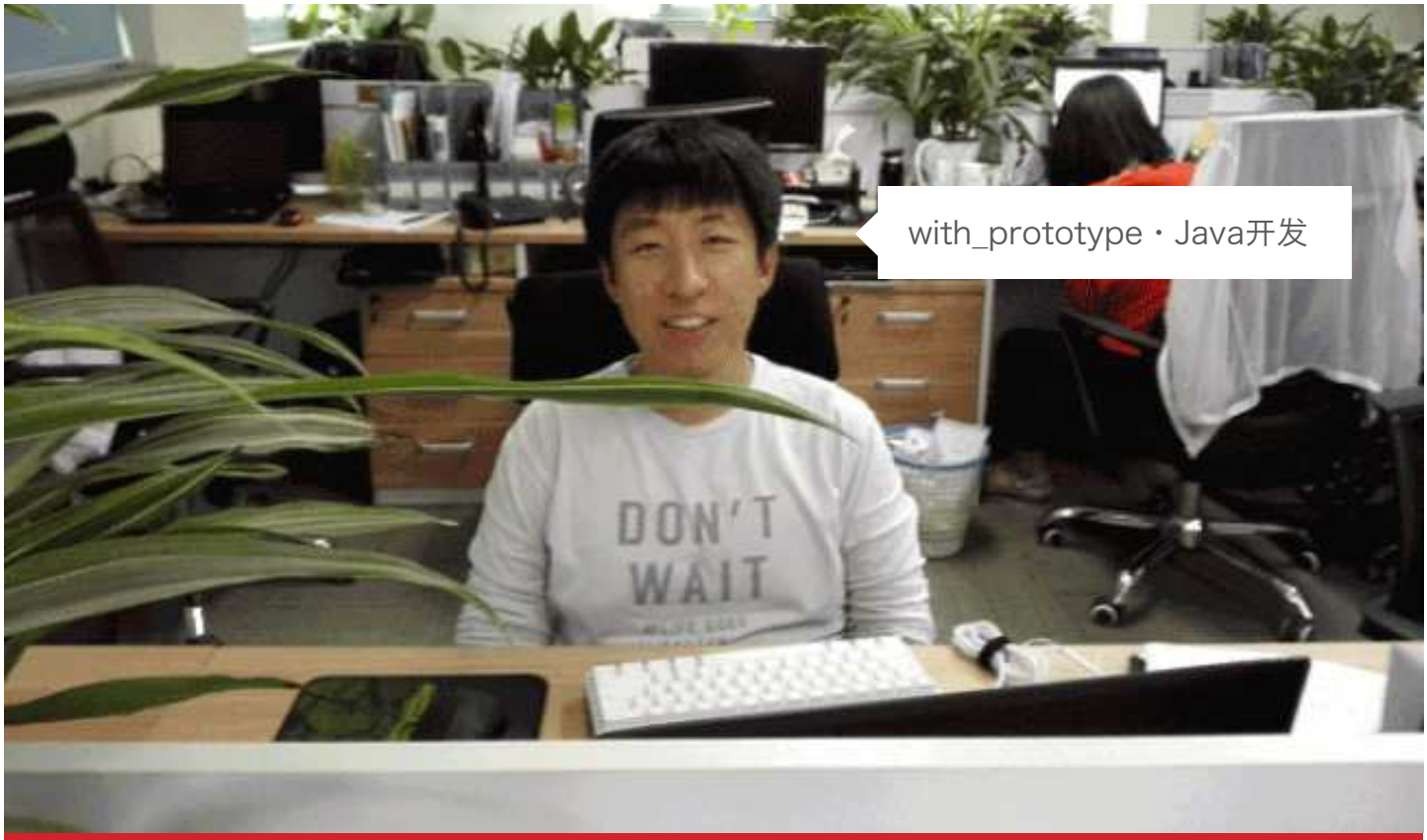
摩诘青溪《西施咏》，潇湘红楼《明妃吟》。

没错，这诗集合了白居易《长恨歌》、曹植《感甄赋》、王维《西施咏》、林黛玉《明妃吟》，椎名的美貌用古往今来的美女诗都搁一块也写不完。淑女椎名倾慕司马雪的才华，从此司马雪正式脱单。

不久，年薪100W的司马雪在北京买房定居。又过了几个月，这对为我国IT事业做出了巨大贡献的情侣在北京举办了盛大的婚礼。

异史氏曰：公司大到一定程度，别有用心的竞争对手就会搞小动作，因此想成为大厂的技术专家，懂点安全方面的知识是很有帮助的。程序员是经常被嘲笑为“注定孤独一生”的群体，但出人头地的程序员抱得美人归的可能性非常大，毕竟女人的颜值通常和她的男朋友的收入成正比。

■



挨踢部落故事汇(23)

一路前行，兴趣所至

with_prototype主要从事Java开发、曾做过配置管理、系统运维、框架研发组件，现在就职于一家保险经纪公司的架构组，担任架构组工程师。闲暇之余，喜欢看技术书籍，偶尔听听音乐，骑骑车，跑跑步，跟朋友玩玩游戏。

兴趣入行赚了第一桶金

with_prototype和很多人一样，不是计算机专业出身，出于兴趣走进了IT行业。追溯到十年前，那时候他上大二，偶然的机会他在校园报刊亭中看见一本月刊杂志“黑客x档案”，他一时间痴迷于杂志中介绍的各种系统漏洞、安全工具、网站提权的知识。于是他买了第一台笔记本，开始实践各种黑客工具。就此与计算机结缘了。

兴趣会使人自发地解决问题。初为小白，新机器很快受病毒困扰，with_prototype无奈去中关村请人重装系统，顺带买杀毒软件。了解到Ghost这一利器，他变身成装机人员。宿舍中，一天三遍刷系统，做镜像，拿同学机器做恢复实验。同宿舍的同学会说他疯了，简直就是褒奖（呵呵）。

就在这学期有C++课程，他看到学院机房里一水儿的IBM工作站，于是找到机房主任，坦白的

说自己喜欢黑客技术，申请当机房网管。机缘巧合，遇到一个病毒分析高手，小一年的学弟，一同管理机房。就这样，with_prototype在机房实践了3年，讨论过如何鉴别病毒，参与过机房布线，做过讲师助手，开发过PHP程序，那是他对程序还不感冒，但程序员的种子已经种下。机房的生活有时候很累，一天都不出屋，遇到机房重做系统，要到凌晨才能完成，但with_prototype觉得很有意思。一年下来他也挣了500块钱，虽然不多，但对于当时的他来说特别兴奋，这笔钱后来全拿去买技术方面的书了。

毕业季现学现卖，培训Java，转型程序员

转眼到了毕业季，那段时间with_prototype很郁闷，学数学专业，但专业受限，路子窄，他又不喜欢。他曾面试过一些硬件公司，最终因专业不对口没去成。也参加过软件公司的校招笔试，写程序拼不过软件学院的同学，一时间不知所措。

身为一个程序开发的菜鸟，with_prototype无法解释市面上各种软件的实现原理，但他认定软件技术和黑客技术同属信息技术的不同分支，相比安全领域寂寞乏味，程序开发无所不能。因此他毅然决然选择Java培训，转型程序员。

那个年头培训机构还不多，没有在线培训课程，他选择了达内。抱着学精的态度，每天朝九晚九，几个月后就出道了，找了份金融行业的软件外包工作。

说道如何快速入门Java，with_prototype觉得主要有两方面：

1、 理解Java核心体系。Java的体系对于初学者过于庞大，通过培训串讲知识点，清晰学习思路，了解编程的套路；

2、 理解编程思想。掌握了Java基础，还要懂得面向对象思想，才能用Java的方式做分析、设计。

举个例子：

with_prototype刚刚学会向上转型时，就是搞不明白为什么要使用内部类。当时这个阶段，花了不少时间，因为语法上都能看懂，但就是不明白含义。其实内部类就是为了完善Java多继承体系而存在的。通常结合代理、接口、内部类的多态机制，实现上层对象对实现细节的忽略，达到面向对象编程的目标。

最快入门Java的方法应该是，自学Java的语法基础，拿上《Java核心技术》，找个内行人划重点、串流程，通过Java核心技术学习面向对象思想，多写多查多上网看资料。随后学Java-web，一定要自己做一个项目，把问题都解决了就通了。

在入门Web程序开发后，他的程序员生活开始了。

工作中坚持学习，兴趣使人成长

现今SOA大行其道，而在with_prototype刚起步的时代，Java提倡的是一种面向配置、面向接口编程。典型的技术框架就是Spring和IBatis。他刚接触配置开发不久，公司大项目就提上了日程。他越想快速搞清楚如何实现配置，并同时完成开发任务，工作的压力就越大。由于对新知识掌握不够牢，他时常感到恐惧。有时候数据库设计不合理，导致工作重复做，也很受挫。

正是在这种情况下，为了缓解压力，with_prototype养成了一些学习习惯。完成日常工作之后，额外花一些时间学习，学习一些与工作无关的开发。多一份了解，就多一份自信。

他做了很多小程序，比如：

1. 用JavaScript操作dom，结合HTML表格，那时候with_prototype不懂样式，他就用table、button标签，实现动态表格，提交修改到数据库；

2. 实现交易撮合算法，模拟股票买卖交易，用Java实现；

3. 用commons-compress写一个文件压缩器，自动备份每月的工作资料、学习设计模式等等；

系统发布那天，他准备好数据库脚本，提交了程序，保障系统顺利上线了。总之，学的多，心不虚。会的多，胆就大。

兴趣导向，在坚守中求变

后来with_prototype换了工作，在保险行业的做运维开发，既兼顾配置管理，又带实习生，在这儿度过这几年，他觉得最宝贵的经历，就是跳出Java日常开发的圈子，掌握了一部分解决问题的能力。运维工作特别锻炼人的思维，分析问题的过程，都是对逻辑思维的锻炼。修改程序逻辑只是一方面，对问题数据的维护，要经过仔细的思考推断，还要能够从多个角度看待问题。

1、学会多角度思考问题

有一次with_prototype拿到一个性能问题，一段程序使用嵌套递归的方式，计算一组继承类的树形结构上各个节点的属性值，从根节点开始，通过配置控制节点计算顺序，先计算根节点，完成一层继承结构的对象计算后，再进行下一层计算，耗时很久。分析程序逻辑是正确的，而问题最后的解决，是推测获取数据库链接过于频繁，加之数据量大，导致数据库连接频繁创建，造成性能问题。他考虑到只是个小模块，采用创建集合缓存数据库连接的方式，解决了计算过慢的问题。后来系统改进，重写了流程控制部分，并且使用了连接池。可见，过度设计有时候会适得其反。

2、自学工具解决自动化问题

在公司Windows服务器上，有很多备份工作需要手工完成。当时的方案是用cmd脚本完成自动化。

学习cmd，是个体力活。这方面相关资料少，除了命令说明，就只能从网上东拼西凑一些资料，自己摸索。但学习的过程是很有成就感的。

with_prototype实现了系统目录的增量备份，结合WinRAR实现自动归档，清理历史包。后来用的熟练了，有很多图形界面工具的命令操作行业可以加入进来。比如VSS管理（定期分析重建索引）、SVN增量备份（每次需要记录上次备份的版本号，就写在一个空文件里，下次继续更新，直接读取文件获取版本号）。

with_prototype这份工作坚持了5年多，周围的伙伴换了一茬又一茬，后来还做了核心系统迁移，通信平台的设计开发。不过，一旦工作变得机械化，乐趣就少了，做来做去，还是跳不出公司部门规划这个圈子。所以他又换了工作。

走进互联网，用开放的技术方式，解决新问题

再后来，with_prototype来到一家互联网公司做研发。在开放自由的环境中，开发人员最快乐的，莫过用更多的方法，解决遇到的问题。可以根据需要，更多的去用开源技术，自己去封装工具。with_prototype前段时间做过一个社交平台，他考虑检索的内容频度、数量大，结构相

近，在设计之初，就考虑将搜索引擎与数据库分离存储业务数据。最终，在lucene上，封装常规索引操作的API，能够更方便的添加索引，查询、按权重排序。完成了主要部分开发。与此同时，他头一次接触到微信、支付宝、其他第三方支付平台，通过抽象不同接口，添加独立配置的方式，实现了支付的核心。之后使用模板引擎，完成了通信内容的转换，一个简单的支付平台也成型了。

健康观念的改变

“脸皮厚，身体好”是with_prototype以前一个部门经理说的，现在他也做到了。工作之余，with_prototype也注重加强锻炼身体。最有效的锻炼，还是跑步，从一开始跑不动，到能慢跑一小时，坚持下来就好了。他偶尔也学学烹饪、烘焙。想减肥，就得少吃油。总有人调侃他为什么不着急找女朋友，其实，黑程序员的人太多了。如果看到喜欢的女孩，多了解，有意识的让自己变得更好，多用心传达……水到渠成是早晚的事儿。

【写在最后】

生活还得继续、工作也不能停，接触更多新事物。现在很难说一个IT人员，能够从业多少年，因人而异，兴趣使然。用心坚守，总会发现新鲜事物，这是一个发展的行业，让每一天都过得有意义，只要兴趣浓厚，即使悲伤、难过，也都是暂时的。希望自己的认识能够对后辈有所帮助，总结以下三点：

首先，要相信自己的选择：从开发到运维，从产品到测试，从安全到大数据，从金融到互联网，从医疗到电商。各行各业，各类技术，心有多大，世界就有多大。

其次，跟上时代的脚步，不断学习，积累沉淀：没有IT基因，没关系，因为再年轻的专家也需要经验的积累，而40岁同样可以学习编程。

最重要的一点，工作再苦再累，都要热爱生活：因为有一种温暖，叫关怀；有一种勇气，叫责任，有一种纯粹，叫爱。

■

挨踢部落故事汇(24)

扎根技术圈归属感爆棚

本期主人公小星星坐标广州，一个非常有活力的城市。他几年前的一次弥留，居然从Flash动画开始了他的代码人生，误入了IT部落。后来的几年，他做过VB桌面程序，做过复杂的C/C++算法，也写过浩大的Java门户，还玩过混合编程即AppCan，现在操着一手世界上最强大的语言PHP（^_ ^），哈哈，就是酱紫。回顾下这些时间的经历，他最深入的算是Java语言Android相关的东西，可谓他的编程巅峰；小星星涉足时间最长的则是PHP，毕竟现在系统是用它构建的；而最有意思最刻骨铭心的，是使用AppCan的那段时光！

编程意识萌芽

小星星第一份工作离职后，他跟团队接外包项目做了一段时间，其中很大一部分是微信端的应用，因为是生手，对HTML5和CSS3相关的东西特别纠结，可以说几个项目下来人都减了一圈（嗯，本来已经是大瘦子了），感觉特别烧脑。不过他熟悉了移动App开发需要考虑各种适配、各种兼容，异步请求是改善交互的很好的方法，交互中数据安全该做哪些操作等等，收获还是挺大的。之后小星星在面试一个开发工作的时候，被问到一个有关移动端开发的趋势问题，他认为“基于H5进行跨平台App构建会是一个趋势”，那

时候还没了解到有关Hybrid App的任何信息（嗯，信息滞后或者说认知缺失非常可怕，不过项目各种折腾下来还是有这种觉悟），他还是对自己的“预见”感到赞赏！

App定制组件开发思路

当外包项目做得水深火热的时候，负责人突然丢过来一个系统，这是一个非常“庞大”的系统，至少在那时他的团队看来，它，非常地复杂（到了修改个菜单栏选项直接花了三天时间的地步，是的，非常恐怖）！然后，非常地荣幸，接手这项目的重任放到了小星星肩上，结果是大年三十和大年初一他都接到了催进度的电话，2015年的春节他过得很特别，至此小星星跟那个系统结上了仇，复杂不说，关键本身还超多不能理解和接受的各种Bug。没想到泪奔的东西还在后面，那个“可爱”的系统，他接手之后直接做到了现在，没错，是现在！

系统纠结归纠结，但它毕竟能给小星星的团队带来营收，它也需要一个移动端。愁于开发资源非常稀缺不能走常规道路，于是他对混合开发的平台和工具各种物色，最终他的团队选择了AppCan。

由于有H5的经验，他担起了基于AppCan的App构建工作。准备工作是官网文档各种翻、IDE各种尝试、官方交流群各种加，反正就这么开干了，那时候是7月份，广州正热，小星星也做得热火朝天。由于前面花了时间研究了官方文档，他在构建App的时候并没遇到太大的困难，基本都能自己消化。难点在于自己App需要一些定制性特别强的组件，比如滑块，比如进度条，比如图片裁剪，比如折线图，这些都是统一的前景和背景，统一的字体和线条风格，用现成的是不可能的只有自己造轮。从简单的入手，第一刀是个进度条，如下：

这个当时做的思考是分解，将它拆成文本提示、前景、背景以及右边的状态这几个部分，每个部分都用div实现，并且他还想实现复用，需要的时候new一个实例，设置些参数就能出来这么个东西。说干就干，思路简单，实际也很简单：

1）这是组件目标位置：

```
<div id="my-content" class="ub ub-ver ub-con">

    <!--Part1-->
    <div class="ub ub-ver ub-ac">
        <!--自定义进度条-->
        <div id="my-progress"></div>
        <!--自定义进度条 End-->
    </div>
    <!--Part1 End-->
```

2）引入自定义css和JS：

```
<link rel="stylesheet" href="../css/appcan.myprogress.css">

<script src="../js/appcan.myprogress.js"></script>
```

3）实例化：

```
//进度条
var myProgress = new MyProgress({
    selector : "#my-progress", //选择器
    type : 0, //样式1
    max_val : 3, //最大值
    cur_val : 0, //当前值
    bg_color: "rgba(22,48,66,1.0)", //背景色
    fg_color: "rgba(99,194,123,1.0)", //前景色
    height: '0.7em',
});
myProgress.set(1);
```

用法就是酱紫，嗯，还算优雅，这里做了样式的扩展入口，最大最小值的定义，前景背景的定义。

具体实现思路是：背景固定宽度和高度，前景与背景同高，靠左跟背景重合对齐，前景设置高亮颜色，宽度由进度指定，根据进度占比确定前景在背景中的宽度占比，这就实现了进度展示，另外展示文本需要设置大一点的z-index属性，在背景上方上下左右居中，最右侧的状态原理相同，不过是上下居中靠右放置，附上实现代码，点此下载extra目录（嗯，以文件的方式，代码太多不贴这里了，另外还有其他几个组件，都是基于AppCan3.2版本构建的，至于现在的新版本是

否适用，小星星就没验证了）。

技术圈子找到归属感

这是小星星在H5方面的第一个组件封装，成就感满满，好像当时某混合开发群里有人问到进度条组件类似的问题，于是他很乐意地将自己的“研究”分享了出去，能够帮助他人，这也是自己的价值体现。分享，也是很开心的事，接下来的时间里将他封装的几个组件都跟大家进行了分享。逐渐地，他在群里成了活跃分子，某天心血来潮，受大家影响（大家都是很萌很可爱的人），他将自己的备注改成了“广州-小星星”，就是技术交流群里很多人认识的小星星。

在群里打拼的日子里，他认识了很多码友，实话说这是很开心的一个团队。在某月的某一天，技术群里的官方代表邀请他加入管理团队，理由是他比较活跃，问了具体事项之后他顺利入围，至此，小星星在技术交流群扎下了根，更多的是因为这个团队，同时还收到了一份特别的礼物。

技术交流群从活力的角度来考虑，这个团队确实可以托付，且不说它背后代表的技术方向，大家从不同地域不同公司不同行业汇聚于此，一起讨论一起解决问题一起提出改进建议，这本就代表着顽强的生命力，它活着，可以说代表的技术就活着。IT行业，保有一个有活力的团队，这就有坚持做大做强的理由和动力！

小星星回顾走过的坑，可以用分解、总结，记录、分享这四点来归纳。

1、分解。开发过程中遇到的问题有难有易，在遇到一个棘手的问题时需要做的是将其分解，比如一个复杂的UI组件可以分解成多个简单的div块，一个复杂的功能可以分解成多个简单的功能，即复杂化简，做到了这点就是自己能力的提升。

2、总结。而在解决问题的过程当中，适时回过头来做些总结，不仅能巩固自己解决问题的思路，还能找到处理问题时不妥或者不够好的方式，从而归纳总结出更好的处理或者改进方式，这也是一个能力提升的过程。

3、记录。记录还有一个好处，详细的有条理的记录，还是他人处理相同或者类似的问题时一个很好的参考依据，这是分享，从这一角度，如何能够更好地将自己的东西分享出去而尽可能地减少他人的疑问，这又是一个功能规划和信息整理的一个提升。

4、分享最重要的，也许你的生活里需要一个“小星星”，成长的过程当中总有不如意，有个精神支柱，或者说有个价值标尺，逆境时不至于让自己一蹶不振，顺境时不至于得意忘形，这道理大家都懂！

生活会齐酸甜苦楚才有意义，曾经有几个月，小星星过得有点困难，与此同时，他所在的团队也过得有点困难，正因为有技术群里这些人的存在，他才有精神的支撑走了过来，小星星团队的领导也因为他和同事们的坚持吃定定心丸，带领团队走过了困难期。感谢有你们！

小星星是个爱分享的小星星，自己写的很好的或者感觉不错的demo，都会整理打包出来，有需要的盆友可以直接拿去用，这是其一；另一个，在整理的过程中他还能够找到那些不太和谐的因素，比如引入的资源文件是否足够简洁，实例化过程是否足够方便，参数是否足够明确，说明文档是否足够直白简单。这些都是使用者需要的，也是对他自己做类似封装的一个方向指导，这无形当中对他的思维、编码技能都是很好的提升。完成一个项目回过头来总结和记录，是对自身和他人成长都很有价值的事情。 ■

小妖・Android开发

挨踢部落故事汇(25)

剖析链式存储栈与代码实现

小妖毕业于重庆邮电大学计算机系，目前是一枚小小Android移动应用开发程序员，爱生活，爱女朋友，没事的时候还爱玩儿游戏，爱死磕。目前人生目标：赚钱买房买车娶媳妇儿。

栈和队列介绍

学过计算机编程基础的朋友都知道计算机中最常见的数据结构栈和队列，这种线性表结构之所以很常见，正是因为它们简单却强大的功能特性是很多算法的基础构成。大三时小妖在北京一次实习面试中，面试官问过他一个问题，“怎么样用栈来实现一个队列？”当时他陷入了一个误区，认为面试官说的栈是指一个栈，由于栈的先入后出的特性，所以一个栈来完成队列的功能在概念上是不可能的。俗话说，一个栈不行，那就两个栈。OK，答对了，确实是至少需要两个栈。言归正传，既然栈和队列的应用如此广泛，那么怎么通过代码来实现他们呢？很多童鞋可能会说，这还不简单么？用集合或者数组，so easy！那么，如果为了提高执行效率，不允许使用集合或者数组呢？下面，就由小妖来为大家小小的剖

析一下吧！

链式存储栈分析

栈的概念，这里就不再多说了，不清楚的童鞋请自行百度。其先入后出的特性确实有很多的实现方式。如题，我们这里将实现链式存储，何为链式存储呢？如下图：

链式存储

链式存储的基本单位是Node（结点），在一个结点中，分为数据区和指针域，通过指针域指向下一个结点来将各个结点相连接。由于链式的特性，插入和删除链中的某一个结点（除链头和尾之外的其他结点）相对而言是消耗比较大的，但是用来实现栈或者队列是极好的。对于栈结构来说，数据的入栈和出栈以及判断是否到达栈底等方法是必不可少的。

代码实现

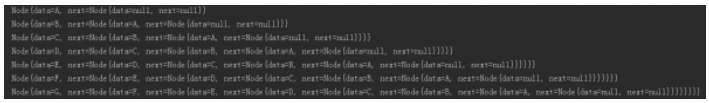
接下来看看如何用Java代码实现：

```
1.   /**
2.    * Created by lxh on 2017/8/1.
3.    * QQ-632671653
4.    * 自定义链式存储栈
5.    */
6.   public class LinkStack<T> {
7.
8.        private Node<T> top = new Node<>();//初始化栈顶结点
9.
10.      /**
11.       * 数据入栈操作
12.       * @param data
13.       */
14.      public void push(T data) {
15.          top = new Node<>(data, top);
16.          System.out.println(top.toString());
17.      }
```

```
1.   /**
2.        * 数据出栈操作
3.        * @return
4.        */
5.       public T pop() {
6.           T result = top.data;
7.           if (top.hasNext()) {
8.               top = top.next;
9.           }
10.          return result;
11.      }
12.      //使用内部类构建结点
13.      private class Node<T> {
14.          T data; //使用泛型来定义数据域
15.          Node<T> next; //定义下一个结点
16.
17.          /**
18.           * 无参构造方法中初始化栈底部，为栈添加末端哨兵
19.           */
20.          Node() {
21.              data = null;
22.              next = null;
23.          }
24.
25.          /**
26.           * 通过带参构造方法来构造每一个添加的结点对象
27.           * @param data
```

```
28.          * @param next
29.          */
30.         public Node(T data, Node<T> next)
{
31.             this.data = data;
32.             this.next = next;
33.         }
34.
35.         /**
36.          * 判断是否到达栈底
37.          * @return
38.          */
39.         boolean hasNext() {
40.             return data != null && next
!= null;
41.         }
42.
43.         @Override
44.         public String toString() {
45.             return “Node{“ +
46.                     “data=” + data +
47.                     “, next=” + next +
48.                     ‘}’;
49.         }
50.     }
51.
52. }
```

从代码中可以看到，在这里使用了泛型来保证入栈数据类型的通用，结点采用了内部类的方式来构造，因为其私有性，所以是可以保证数据安全的。基础不太好的同学可能会比较懵逼，不太明白这个链式是链到哪里的，所以小妖在入栈操作时将栈内信息打印了出来，相信各位一看就明白了。如下图：



相信通过信息图大家明白了，这个链式存储栈是通过在结点对象中递归添加结点对象来构成了一个一个结点的链接，体现在字符集上则是一层一层的对象嵌套（由于采用了Java语言来编写，没有C中的指针，所以采用此方法来代替指针）。通过末端结点的哨兵，当出栈时数据集为null 则可以判断出栈中数据出栈完毕了。

【写在最后】

同行中流传着这样一句话：不要重复的造轮子。但是小妖觉得，即便不需要你去造轮子，但是你也得明白轮子到底是怎样造出来。如果你只会调用API，那么你将失去竞争力。

■

挨踢部落故事汇(26)

全栈攻城狮之葵花宝典

曾在移动、前端、后台都有涉猎的小星星，不说十八般武艺样样精通，也是熟知通晓，运用自如。而今回顾他的IT坑，将成长的经验分享给大家。从JS到Java再到PHP，面向全栈的攻城狮已经上线。

小星星的入门语言是AS，没错，就是Flash脚本语言。从偏门闯入了这浩大的IT世界，于是他会了JavaScript 、HTML、CSS、C/C++、Java、PHP等语言。Flash追求美观和谐，它总能给人一种美的享受，于是“爱美”的这种习惯一直保持，以至于后来对于其他语言的学习都会各种纠结与格式化，慢慢练就了洁癖的本领。没错，就是洁癖！

萌芽意识

初识ActionScript，是为了满足Flash场景中一个交互按钮的响应。为了加上onclick这个事件，小星星记得那时候足足花了半年的时间（自嘲下，那时候就是那么地，艰苦！），在看到提示信息出现的那一刻，整个人几乎激动到跳起来，这是多么的来之不易。自此他意识到了“对象”的概念，即真实存在的物体，或者称之为Object。

有了对象，就可以把很多重复的图层元素归并到一起，很多场景操作就好办了，具体操作方式就是在时间轴上抽象出各种组件，通过组件组织成各种场景，抽象的最大好处就是可以复用，降

低场景上下文之间的耦合，这样也使作品有更好的可维护性。

自我挣扎中成长-发现瑕疵

小星星接触了JavaScript 之后，发现语法上它跟AS惊人地相似，不奇怪，后来了解到它们的核心都是ecmascript，也罢，入门JavaScript 几乎没花什么力气。在借助AS拿下JS常用的半边江山之后，发现某些功能的某些写法实在是冗余，或者说不够干脆。比如：

```
1.   var hasContent;
2.   if(arr.length > 0){
3.       hasContent = true;
4.   }else{
5.       hasContent = false;
6.   }
```

其实可以很简单的一个句子处理好：var hasContent = arr.length > 0; 当然如果要严谨一点，这里还有一个定时炸弹，即arr这个变量为null的时候会报错无法继续，所以完善点应该是：

```
1.   var hasContent = arr != null && arr.
length > 0;
```

上面这种是很简单的一个变量判断与赋值，完全可以写得很简洁。另外是变量与方法命名，有些人喜欢驼峰法命名，有些人喜欢小写字母加下划线，这些都没问题，统一的写法看起来会让人感觉很舒服，但往往有一些看起来就狠抓狂了，比如上面这个例子用他们的写法：

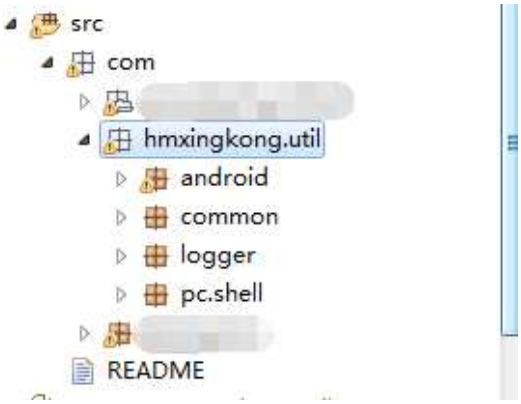
```
1.   var a;
2.   if(b.length > 0){
3.       a = true;
4.
5.   }else{
6.       a = false;
7.   }
```

这种代码的可读性可以说非常地差，每看一行几乎都是用猜的，上面这个还比较容易猜，如果篇幅再大一点，那看的人就会感觉要吐血。

所以，一段代码或者一个程序，从编码角度来讲是马是骡，看它是否整洁是否好理解就一清二楚，当然健壮性也很重要，总不能它跑着跑着就异常挂了吧。于是乎，只要接纳第三方代码进入小星星的作品，他都会对代码进行整改，让它变得跟自己的原有逻辑不那么格格不入，当然作者还是署名原作者。这么下来也可以说练就了自己的一双火眼金睛，别人的代码哪里可能有问题一看就清楚，也算是自我纠结的成果吧。写好代码，为自己方便，也为他人方便。

归零与记录

跨过第一个门槛之后，编码之路会顺畅很多，基本上就不用纠结语法层面的东西。取而代之的更多的是去思考要完成一个功能该怎么去组织代码结构，怎么样做会比较高效和健壮。比如封装，比如分包，比如抽象公共接口，这些都需要有足够编码经验作为支撑。小星星个人的话，因为平时都趋于追求完美，各种觉得好的情况都会尝试，于是有了下面的这个作品，算是他个人从事Java留下的痕迹吧。



各个Java项目积累下来的工具包

是的，这时候追求跨平台，即Android有的东

西都想在PC实现一遍，或者PC有的东西都想在Android实现一遍，这就是纠结，烧脑！好处是在其他编码工作上遇到一些兼容性问题会考虑得比较周全。

这里面就有个有关图像识别的东西，当初好像是因为一个依赖库版本的问题（它，默默地升级了），搞得好好的代码硬是编译不下去，各种尝试之后无丝毫进展。在一筹莫展的时候，小星星翻烂了的度娘在一个非常不起眼的角落给了条信息“xxxxx最新发布版本为v2.4.0”，心如死灰的他重新燃起了希望，马上折腾开来。果然，升级了依赖库之后一切都回归正常，这是个巨大无比的坑，绝对是！当时已是午夜，为了铭记，决定将这坑记录下来，于是结识了ddJava（译为：顶顶Java），坛主为“大猩猩”，当时这是个全新的论坛，几乎捡了个沙发，在此感谢猩猩！

论坛发帖记录问题，同时也是个回顾的过程，这期间不无可能会发现新的问题新的解决问题的方案，从某种程度上来讲，这也是一种学习的过程。解决了问题，再回过头来看，可能会发现更好的解决方案！

编译问题解决了，又遇上了另一个问题。当时的匹配算法比较纠结，只能针对指定区域进行匹配。指定区域就是要确定坐标，做法是匹配传参时，将坐标也一起传过去。小星星这样想也就这样做了，大半夜地在调试，当完成时他跟公司大牛人物提了这事，结果对方来了句：为啥你的坐标不放在图片文件名里！匹配时直接解析文件名，这不是更方便？小星星恍然大悟，确实大神级就是不一样，他们往往能抓住问题的关键点，可以很犀利地处理好问题。这是学习的另一种方式，跟大神交流，分享自己所知，让大神指出改进优化的地方。

逆向思考

当对功能代码已经轻车熟路的时候，你会发现眼前的小问题本不是问题，但无数个小问题放一起形成的大问题却很难去逾越。这就是成长过程中的另一个坎。技能方面已经能驾驭语言的情况下，剩下要思考的更多的问题是系统性能和业务逻辑，就是自己的代码如何完美地配合去完成一个既定业务，又不影响到系统的可维护性和稳定性以及运行效率，可能方法一可行，方法二也可以，方法三也凑合。这时候就需要从思维上上升一个高度，跳出来，从顶层往下看，从需求引导自己的编码。

比如产品需要在某个列表中添加一个搜索选项，而这个选项涉及另外一个或者几个千万级大表（MySQL，是的目前来说它还是非常受欢迎的）。从语法层面，这个问题非常简单，做法是直接连表查询，条件中加入新的过滤条件，这种做法能够实现产品的需求，但带来的问题是使用系统的人会因为慢而抓狂！问题在于大表连表查询，这很消耗数据库的性能。用户需要这功能但TA不会接受由此带来的慢，所以方法一不可行。

方法二有？嗯，毋庸置疑肯定是有的！既然是大表跨表检索影响效率，那就不让它跨表。做法是将目标条件先在其所在的驱动表先过滤一遍获取目标值，再用这个中间的值去主表里过滤查询。相信这个方法也不难，但在数据量上去以后这种做法绝对会比直接连表快上好几倍。

方法三？是的，从更高阶的层面去思考。最彻底的做法是从关系表的设计出发，从根源上尽可能地避免大表的连表查询（嗯，这个还未实际尝试过，事实上它是可行的，也见过某些骨灰级的任务这么做），另外还可以借助服务器的强大的计算能力，将目标数据读入内存在内存中进行条件检索，相信这种做法对于多个大表检索的性能会有指数级的提升。■

挨踢部落 技术坐诊

精选

大数据·云计算·移动

挨踢部落是为核心开发者提供深度技术交流，解决开发需求，资源共享的服务社群。基于此社群，我们邀请了业界技术大咖对开发需求进行一对一突破，解除开发过程中的绊脚石。

提到系统版本，问下，现在安卓开发，一般对老安卓系统的支持到哪个版本？对于3.X的老机器，下载App进行安装的话，一般怎么提示呢？

现在一般是4.0以上，最低到4.0，4.0以下一般都不支持了。安装App一般是进入引导页之后进行系统版本判断，低于4.0的话， 弹窗提示或者吐司提示手机系统版本过低，不能支持当前应用，然后退出程序。

支付宝回调，说签名不对是什么鬼啊？

之前弄过支付宝验签，首先别着急，他那个文档是稍微有点那啥。剩下的就是简化交易字段，检查验签方法匹配，我记得有个两种加密方法吧，别选错了。虽然文档不好，拿过来直接用差一点，但是文档确实没有错，没有少东西。就是，你拼装的不对，就返回这个信息，另外看下订单是否匹配。

大家使用的项目管理系统是自己领取任务的做还是领导分配任务的做？我们和北京沟通用的是jira，现在自己想做一套系统。还有一个问题，一般情况下一个项目工单在某一时刻应该只会有一个执行人，但是对于一个大项目的话，可能需要A、B、C一起进行，这个时候的执行人怎么安排的？还是说将一个大项目分解成N个小的项目工单？

1、如果开发是以KPI为主导的，估计自己领任务的多。

2、我们以前用redmine，是分任务的，主要还是用于对程序员、对测试、对需求人员。可以找bitnami-redmine，安装很方便，流程可以自己控制，类似jira，开源的。这要看你怎么理解“一起进行”，可能是三个人先后串行，也可能是独立完成，串行的情况，可以在任务中描述好，每个人之后的交付人，实际操作这完成后，将任务指派给下一工作者，对于管理人员，把自己挂在任务的关注者上。如果是并行，可以将任务拆分为子任务，先创建一个任务，在任务基础上，创建三个子任务，子任务指派个具体的工作人员，就可以了。

有一道面试题，Java栈大小为3，先入栈1 2 3 4 5 6 请问出站顺序最不可能的是？

A 123456

B 213456

C 342156

D 432156

求大神指点。

1、D 吧，栈后进先出，如果先出4，那此时应该是1->2->3->4，这时候栈大小起码要4。

2、D。栈容量限制。

[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\} 谁可以帮我解释一下查资料没看懂，比如300.300.300.300按照这个正则是不是也符合？这个其实还没有验证ip地址是否要求的 可以这么理解？

1、是的，[0-9]\{1,3\} ，0到9的数字有1到3个。

2、对，这个不一定是符合要求的，我记得ip地址每段的最大数好像不超过255。ip地址过滤，正则表达式，匹配ip，三位数一个点，其中的那三个点就是ip地址的点，就是这个意思，数字是在0-9之间，位数是三位，当然还有0的情况。

3、应该是ip过滤，0-9的数字，1到3位 .0-9的数字，1到3位 .0-9的数字，1到3位 .0-9的数字，1到3位，就是可以是0、10、900，.前后的值是0-999。不能完全保证符合要求，但是保证了是数字，最多三位，最少一位，就是过滤了一些，但没过滤完。

问个AIDL问题，谁比较清楚的，AIDL调用会卡调用的线程么？notify是随机唤醒线程还是唤醒第一个wait的线程？

AIDL调用不是异步的。随机唤起一个等待线程，唤起哪个线程，是由VM选择的，notifyAll可以唤起所有等待线程。

问一下有用过AppCan 4.0插件调用支付宝付款的吗？新版插件支付宝支付的时候，提示的是ali40247。然后我看到生成支付请求 generatePayOrder 返回的sign_type 值是rsa。只有2个动作，一个生成支付请求，一个支付，应该是支付的时候有返回order。

参考这个链接>>，初步看，应该是权限的问题，你先按照这个解决。另外生成的签名有的秘钥是rsa 的，只上传了rsa2的，又补了一个rsa的，最终解决。

各位大神，请教个物联网技术问题？怎么实现数据采集传输到控制台显示出来，时间在一秒以内？现在的时延是3-8秒。Wifi和有线网络都有。有一些是在室外。木有各节点的数据，是测的总时间，掐表算的。

从IOT 开始，到你的路由到你的Server，再到控制台前端，各段的时延分别是多少？整个通路是http还是socket 长连接？在协议中，维护一个timestamp列表，监控看看一下吧，否则神也不会知道你慢在哪。或者把一个请求在各节点的日志汇总一下，通过日志的方式看看吧。实际上是一个全链路监测的问题。

态牛 TECH NEO

挨踢故事汇

开发者分享的职场、技术经验

探 城

探索互联网企业内部技术架构

技术坐诊

帮助开发者解决技术痛点难点

北京无忧创想信息技术有限公司

TEL:68476606-8123

HTTP://WWW.51CTO.COM

EMAIL:LINSS@51CTO.COM