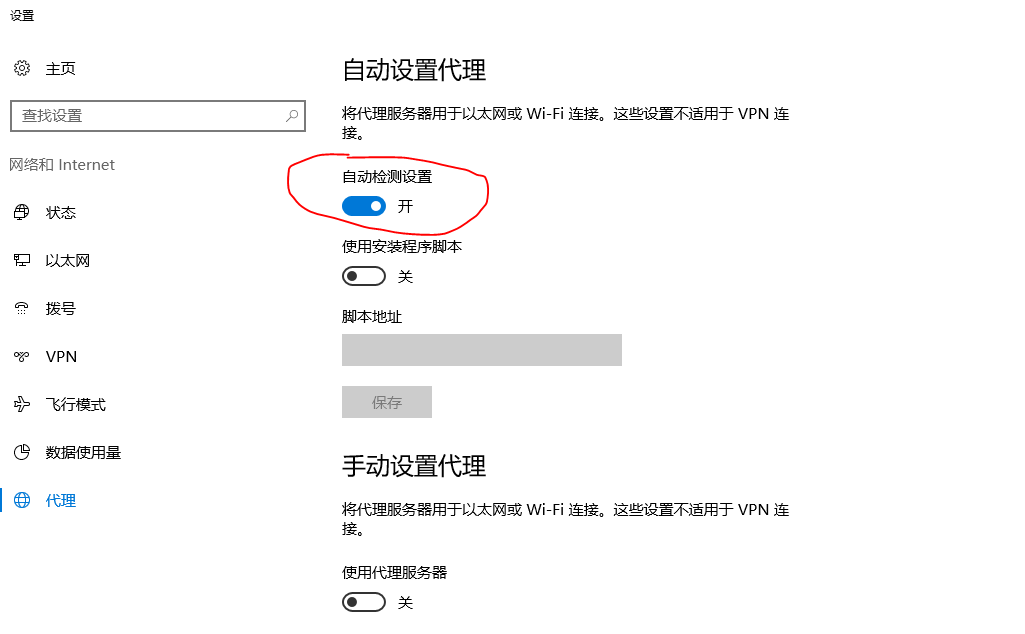
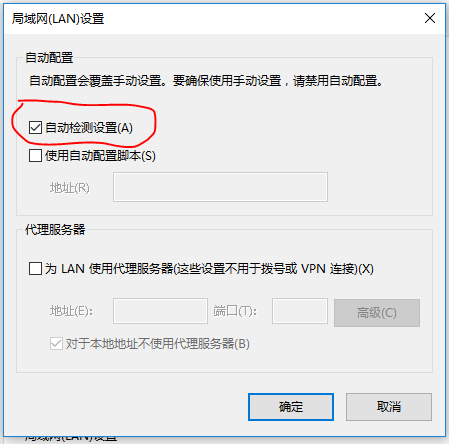
# Tips

1. 登陆jira，开机先要ping 192.168.5.21
2. Eclipse显示红叹号，把JRE系统库的执行环境改成JavaSE-1.8
3. 192.168.10.8X用户名密码
4. root/Bubi@2016
5. 数据库（都在84上)
6. root/bubi@2016
7. 电脑出现可以上网但是网页打不开，解决方法







192.168.10.180：用户名密码：root/Bb#!99

2.1底层账号

{

"error\_code": 0,

"result": {

"address": "bubiV8iF3XMpRf7DuVFbdLbAWNvDRn3qBre9i2Gj",

"private\_key": "privbwQiJqXw28fedN5oEEBLVF8GKhck6WtMMtzNNBHMGLRpGFAjwWJD",

"public\_key": "E4dRXiC9KAwCPEzYf8wP5WeBHRThPU5Fp1kTy1CnEMnr",

"public\_key\_raw": "c215db22c39eb1ed6601e4942cae414fc5606bb6723dc95fad5a3e3a6d755aeb"

}

}

# Linux

1. 出于安全考虑，输入账户密码时字符不会在屏幕上回显，光标也不移动
2. Centos7 连不上网，且ifconfig里没有eth0但是有eno16777736。

需要把eno16777736改名成eth0，具体做法：

<http://jingyan.baidu.com/article/7f41ecec1b022e593d095c1e.html>

/etc/sysconfig/network-scripts/ifcfg-eno16777736 被改成ifcfg-eth0之后还要加上HWADDR这个宏。

1. 守护进程：一般terminal接收一条指令后会等待指令完成才能执行下一条，但是很多服务都是启动一下就转成后台运行的，这种就叫守护进程，

创建子进程，父进程退出

这是编写守护进程的第一步。由于守护进程是脱离控制终端的，因此，完成第一步后就会在Shell终端里造成一程序已经运行完毕的假象。之后的所有工作都在子进程中完成，而用户在Shell终端里则可以执行其他命令，从而在形式上做到了与控制终端的脱离。

在Linux中[父进程](http://baike.baidu.com/view/8507578.htm)先于子进程退出会造成子进程成为孤儿进程，而每当系统发现一个孤儿进程时，就会自动由1号进程（init）收养它，这样，原先的子进程就会变成[init进程](http://baike.baidu.com/view/2965338.htm)的子进程。

查找id

[root@localhost ~]# pgrep -l bubi

2712 bubi

2714 bubid

centos在yum install报错：Another app is currently holding the yum lock解决方法

centos在yum install报错：Another app is currently holding the yum lock，这个问题可能是很多的新手经常遇到问题，之前也有人问我，包括本人在刚刚学习centos的时候也遇到过。

Another app is currently holding the yum lock，这个意思很明显就是说，有另外一个应用在使用yum，被占用锁定了，所以咋办呢，直接结束掉呗。

可以通过强制关掉yum进程：

# rm -f /var/run/yum.pid

然后就可以使用yum了。OK，很简单。

# [centos主机名bogon解决办法](http://www.cnblogs.com/dvking/articles/2559486.html)

bogon是指那些不该出现在internet路由表中的地址吧  
这些地址应该包括：  
1，私有地址如10，172.16-32,192.168.....  
2，还未正式分配出去的地址

经过google找到修改名称的办法

二：修改linux主机的配置文件/etc/sysconfig/network 和 /etc/hosts  
1.修改/etc/sysconfig/network里面的主机名字。  
# vi /etc/sysconfig/network  
NETWORKING=yes  
HOSTNAME=yourname //在这修改hostname  
GATEWAY=192.168.1.1 //你的IP地址，这里我就写的1.1  
2.修改/etc/hosts里面的名字  
# vi /etc/hosts  
127.0.0.1 localhost.localdomain localhost//这个其实没改  
192.168.1.121 yourname //在这修改hostname //你的IP地址  
//有时候只有这一行  
127.0.0.1 yourname localhost.localdomain localhost  
3.让更改的名字在不重启机器下生效  
# hostname yourname //yourname换成localhost  
# su

# BlockChain

安装问题：

Bubi和bubid跑不起来：

1./usr/local/bubichain 里的文件结构缺失，data和log虽然是空的也要有。

std::*map*<std::*string*, routeHandler> mRoutes;是名称和具体回调函数的map

WebServer::Initialize 将名称和回调函数安装到mRoutes里。

安装：addRoute

Webserver initialize函数中先new了一个server(server\_ptr\_), server的构造函数

server::server(const std::*string*& address,

unsigned short port, asio::ssl::context \*context, size\_t pool\_size)

: io\_server\_pool\_(pool\_size)

, signals\_(io\_server\_pool\_.get\_first\_service())

, acceptor\_(io\_server\_pool\_.get\_first\_service())

, connection\_manager\_()

, socket\_ptr\_(NULL)

, sslsocket\_ptr\_(NULL)

, context\_(context)

socket\_->async\_read\_some 中从asio的buffer中读取消息然后转换为request\_在mRouts中找到对应的handler进行调用

do\_read被connection的start调用，start又被connection\_manager的start函数调用。Connection\_manager的start被

权重：

账户A被创建时，可以设置sign1是B，signer2 是A，初始化权重。同时A自己有master权重。

之后可以通过设置帐号权重值这一交易类型进行重新设置。

每个账号有三个权重门限值，high>media>low

其中设置属性需要达到high门限，其他动作需要media门限。

A发行一个资产，privkey中选择了A+B,则验证A的master权重+B的权重>=media

A发行一个资产，privkey中选择了B+C,则验证B的权重+C的权重>=media

# PostgreSQL

只有root有权限su postgres

如果在别的账户下只能sudo su postgres

# MySql

登录：

[root@localhost classes]# mysql -u root -p

Enter password:

（直接输入密码此处不回显）

看数据库：

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| api |

| bbs\_bumeng |

| bubi\_asset |

| bubi\_dev |

| bubi\_operation |

| bubi\_uc |

| monitor |

| mysql |

| performance\_schema |

| publicity |

| sys |

+--------------------+

12 rows in set (0.00 sec)

选一个库，

mysql> use api

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

看数据库里的表

mysql> show tables;

+--------------------------+

| Tables\_in\_api |

+--------------------------+

| b\_asset |

| b\_asset\_his |

| b\_bill\_transaction |

| b\_blend\_opt |

| b\_cas\_hash\_txt |

| b\_cas\_info\_sid |

| b\_evidence |

| b\_evidence\_signers |

| b\_info\_evidence |

| b\_last\_tx\_data\_4platform |

| b\_request\_api |

| b\_sys\_log |

| b\_transaction |

| source\_account |

| trade\_serial |

| tx\_timeout |

+--------------------------+

16 rows in set (0.00 sec)

查看一个表的描述

mysql> describe b\_asset;

+--------------+--------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+--------------+--------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| app\_id | varchar(50) | YES | | NULL | |

| asset\_name | varchar(60) | YES | | NULL | |

| asset\_code | varchar(80) | YES | MUL | NULL | |

| asset\_issuer | varchar(255) | YES | | NULL | |

| asset\_unit | varchar(30) | YES | | NULL | |

| asset\_amount | bigint(20) | YES | | NULL | |

| create\_time | datetime | YES | | NULL | |

| update\_time | datetime | YES | | NULL | |

| hash | varchar(255) | YES | | NULL | |

| icon | varchar(255) | YES | | NULL | |

+--------------+--------------+------+-----+---------+----------------+

11 rows in set (0.00 sec)

NULL一列表示是不是可以留空。

PRI:主键，每一行都有且各不重复的一个值，可以用来遍历。主键不可为空

MUL:可以重复

mysql> describe b\_attention;

+-----------+---------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-----------+---------------+------+-----+---------+----------------+

| id | int(11) | NO | PRI | NULL | auto\_increment |

| uid | varchar(255) | NO | | NULL | |

| node\_guid | varchar(255) | NO | | NULL | |

| node\_name | varchar(255) | NO | | NULL | |

| pos\_x | decimal(11,3) | NO | | NULL | |

| pos\_y | decimal(11,3) | NO | | NULL | |

+-----------+---------------+------+-----+---------+----------------+

6 rows in set (0.00 sec)

mysql> select \* from b\_attention;

+----+----------------------------------+----------------------------------+-----------+---------+---------+

| id | uid | node\_guid | node\_name | pos\_x | pos\_y |

+----+----------------------------------+----------------------------------+-----------+---------+---------+

| 1 | 3bc6e842f63d5e5a35942fac4de8cd01 | 5fb0797c40d4c8194824d346b97ea679 | 1? | 230.000 | 320.000 |

| 2 | 3bc6e842f63d5e5a35942fac4de8cd01 | 7ffab38a22fbb8c1350c0e74334ae5f9 | 2? | 422.000 | 138.000 |

| 3 | 3bc6e842f63d5e5a35942fac4de8cd01 | 2fed668146dfe3d7afc22a33cb369ec2 | 3??? | 183.000 | 84.000 |

+----+----------------------------------+----------------------------------+-----------+---------+---------+

3 rows in set (0.00 sec)

mysql> select id from b\_attention where node\_name = "1?";//选的是id那一列

+----+

| id |

+----+

| 1 |

+----+

1 row in set (0.00 sec)

mysql> select \* from b\_attention where node\_name = "1?";//选的是一整行

+----+----------------------------------+----------------------------------+-----------+---------+---------+

| id | uid | node\_guid | node\_name | pos\_x | pos\_y |

+----+----------------------------------+----------------------------------+-----------+---------+---------+

| 1 | 3bc6e842f63d5e5a35942fac4de8cd01 | 5fb0797c40d4c8194824d346b97ea679 | 1? | 230.000 | 320.000 |

+----+----------------------------------+----------------------------------+-----------+---------+---------+

1 row in set (0.00 sec)

# C++

实际上，成员函数默认的第一个参数为T\* const this.

void WebServer::SubmitTransaction(const http::server::request &request, std::*string* &reply)只有两个参数，可是bind的时候

server\_ptr\_->addRoute("submitTransaction", std::*bind*(&WebServer::SubmitTransaction, this, std::placeholders::*\_1*, std::placeholders::*\_2*));

bind 第一个参数 函数名，后面是每个参数，如果在bind的时候不能确定参数是啥，就用placeholders

通过std::function对C++中各种可调用实体（普通函数、Lambda表达式、函数指针、以及其它函数对象等）的封装，形成一个新的可调用的std::function对象；让我们不再纠结那么多的可调用实体。一切变的简单粗暴。

std::function对象最大的用处就是在实现函数回调，使用者需要注意，它不能被用来检查相等或者不相等，但是可以与NULL或者nullptr进行比较。

单例模式：

public class Singleton {

    private static Singleton singleton = null;

    private Singleton() {  } //构造函数是私有的

    public static Singleton getInstance() {

        if (singleton== null) {

            singleton= new Singleton();

        }

        return singleton;

    }

}

Cpptest新加入了http\_client源，显示各种undefined reference 连接错误，解决：在makefile.am中加入

cpptest\_SOURCES = ../../src/utils/file.cpp\

../../src/utils/logger.cpp\

../../src/utils/net.cpp\

../../src/utils/thread.cpp\

../../src/utils/timestamp.cpp\

../../src/utils/utils.cpp\

../../src/utils/crypto.cpp\

../../src/utils/lrucache.hpp\

../../src/utils/noncopyable.h\

../../src/utils/timer.cpp\

../../src/utils/system.cpp\

common/web\_socket\_server.cpp\

common/websocket\_test.cpp\

common/http\_client.cpp\

main.cpp

# JAVA

Java和c++优缺点

1，最大的区别：比较明显的是C++能被操作系统（Window，Linux）直接运行，而Java不行，需要有个JVM.也就是说，java编译好的文件需要通过jvm来让操作系统运行。jvm可以安装到任何系统。  
  
2：JAVA的应用在高层，C++在中间件和底层……   
JAVA离不开业务逻辑，而C++可以离开业务为JAVA们服务……   
JAVA的一堆框架，一堆中间件，我们都只能跟在后头，除非你是greate master!而C++可以创造出属于自己的东西，尽管不是那种greate master.   
  
3：还有一点就是效率问题：JAVA更注重于[面向对象](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWI-Pjmzn1nknjIbnvD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkn161PHcvnW0)的思想(为了完全[面向对象](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWI-Pjmzn1nknjIbnvD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkn161PHcvnW0)，可以适当的牺牲效率)，面向对象使复杂的事情简单化  
而C++更注重于效率(为了更好的效率,可以适当的牺牲面向对象) ，所以C++更复杂一些。

因为c和c++是直接编译成可执行文件，所以运行效率要比java高

java因为是运行在虚拟机上，不需要考虑[内存管理](https://www.baidu.com/s?wd=%E5%86%85%E5%AD%98%E7%AE%A1%E7%90%86&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPvn1n1nLuHwbnH7hnHfv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3n1RzPWcs)和[垃圾回收](https://www.baidu.com/s?wd=%E5%9E%83%E5%9C%BE%E5%9B%9E%E6%94%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPvn1n1nLuHwbnH7hnHfv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHn3n1RzPWcs)机制。

java中有一个根类object，所有的类都是其子类，通过这种方式将容器和算法分离，实现一种操作作用于多种对象，提高代码重用。

c++中没有总根对象，但是c++提供了另一个更强大的功能“模板”，同样高效地实现了一种操作作用于多种对象，提供了高效的代码重用方法。  
6. [数据结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWI-Pjmzn1nknjIbnvD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkn161PHcvnW0)：java内建了丰富的[数据结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWI-Pjmzn1nknjIbnvD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkn161PHcvnW0)：列表，集合等等（很久没用java了，有些记不太清）。而c++则用“模板”同样提供了各种[数据结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dBPWI-Pjmzn1nknjIbnvD0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkn161PHcvnW0)（容器）。

Object是所有类的父类

Final关键字用来限制其不会被子类改写。

New出来不用delete

在[JAVA](http://baike.baidu.com/subview/29/12654100.htm)中：类名的标识符一般用大驼峰式书写格式，方法和变量的标识符则多用小驼峰式书写格式

Java 数据类型被分为：基本数据类型和引用数据类型。

基本数据类型：boolean,byte,char,short,int,long,float,double

其他都是引用，相当于指针，用的时候要先new出来。

Java 的常量用 final 说明，约定常量名一般全部使用大写字母，如果是多个

单词组合在一起的，单词之间用下划线连接，常量在程序执行时不可更改。

如：final int i=1;

i=i+1; //错，i 是 final（常量），不可更改值

例如：final double IP = 3.14159 D

说明：默认类型为 64 位 double 双精度类型(D 或 d),数字后面加 F 或 f 则是

32 位 float 单

精度(实数)类型

Switch(expr)

expr 必须是与 int 类型兼容的类型，即为 byte，short，char 和 int 类型

中的其中一种

多重break实现：break/continue+标号

ok:  
 for(int i=0;i<5;i++)  
 {  
 for(int j=0;j<10;j++)  
 {  
 System.out.println("i=" + i + ",j=" + j);  
 if(j == 5) break ok;  
 }  
 }

可以直接break出ok下面的for循环，标号必须在for循环或while/switch上面。

int a[][]=new int[2][] ; a[0]=new int[3] ; // 第二维第一个元素指向 3 个整型数

a[1]=new int[5] ; // 第二维第一个元素指向 5 个整型数

注意：Java 可以第二维不等长

JVM GC是JVM的内存回收算法,调整JVM GC(Garbage Collection)

匿名对象：

method(A a);//a是A类型的对象

method (new A());//A类型对象被new出来但是没有名字，所以叫匿名对象

一个类在定义时，如果没有定义过构造函数。那么该类中会自动生成一个空

参数的构造函数。为了方便该类创建对象，完成初始化。

如果在类中自定义了构造函数。那么默认的构造函数就没有了。

简单一句话：你写了，默认就没有了。你没写就只有默认的。

一个类中，可以有多个构造函数，因为它们的函数名称都相同，所以只能通

过参数列表来区分。所以，一个类中如果出现多个构造函数。它们的存在是

以重载体现的。

.java为Java的源文件后缀，编写的代码需要写在.java文件中。  
而.class是字节码文件，是.java源文件通过javac命令编译后生成的文件。  
Java虚拟机就是去运行.[class文件](https://www.baidu.com/s?wd=class%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4Pvc1mvDYnymLrAw-mHn0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtzPWR4nWR3rjD)从而实现程序的运行。

通俗的说，  
包就是包括文件的文件夹  
例如import [java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4mhmsnHNWnyDYnhn1nvR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtknWn3nHfYPWbsrjc4rHDdrjb4).\*;  
这里导入的util文件夹，里面所有的类是不是全部用到暂且不论  
而类，就是文件夹中的文件  
import [java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4mhmsnHNWnyDYnhn1nvR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtknWn3nHfYPWbsrjc4rHDdrjb4).Date;  
这里明确指的就是Date这个类，是[java.util](https://www.baidu.com/s?wd=java.util&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4mhmsnHNWnyDYnhn1nvR0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtknWn3nHfYPWbsrjc4rHDdrjb4)这个包里面的类。因为有许多的包同样有Date为名称的类。  
如import java.sql.Date；在这里也有一个

Public 其他包也可以访问

Friendly（啥都不写）只有本包可以访问

super，代表父类。

1. **class** cam0
2. {
3. **double** size=10;
4. **void** printsize()
5. {
6. System.out.println(size);
7. }
8. }
9. **class** cam1 **extends** cam0
10. {
11. **double** size=20;
12. **void** printsize()
13. {
14. System.out.println(size);
15. System.out.println(**super**.size);
16. **super**.printsize();
17. }
18. }

成员变量和静态变量的区别：

1，成员变量所属于对象。所以也称为实例变量。

静态变量所属于类。所以也称为类变量。

2，成员变量存在于堆内存中。

静态变量存在于方法区中。

3，成员变量随着对象创建而存在。随着对象被回收而消失。

静态变量随着类的加载而存在。随着类的消失而消失。

4，成员变量只能被对象所调用 。

33

静态变量可以被对象调用，也可以被类名调用。

所以，成员变量可以称为对象的特有数据，静态变量称为对象的共享数据。

优先级由高到低依次为：this.show(O)、super.show(O)、this.show((super)O)、super.show((super)O)

2017.2.28 bumeng上线bug：

用户无法添加节点，后台报错“无权限”

原因：Long型比较应使用equals，不能用==。

Long是对象型，相当于c++指针，==验证的是指针的值相等，equals是指向的数。

任何时候==都是比较的本身，基础类型==是他们的值，引用类型==是他们的地址。（引用类型定义的时候需要new出来否则就是空的？）

Equals():如果equals没有重载过，比如object类型里面的equals，他的实现是return this==obj，所以和==是一样的，但如果重载过，一般会变成地址里的值的比较，而不是地址本身。

C++重载和默认参数二义性

C++允许重载（参数列表不同，比如参数列表是一个int还是两个int，只有返回值不同不能当成重载，会报错。）

也允许默认参数，（int a ,int b = 0）这种。

当出现歧义时编译器会报错的。

在java里，没有默认参数，只能通过用重载调用this.方法名（传参，传参，固定值，固定值）来实现。

# [Integer与int的种种比较你知道多少？](http://www.cnblogs.com/liuling/archive/2013/05/05/intAndInteger.html)

　　如果面试官问Integer与int的区别：估计大多数人只会说道两点，Ingeter是int的包装类，int的初值为0，Ingeter的初值为null。但是如果面试官再问一下Integer i = 1;int ii = 1; i==ii为true还是为false？估计就有一部分人答不出来了，如果再问一下其他的，估计更多的人会头脑一片混乱。所以我对它们进行了总结，希望对大家有帮助。

　　首先看代码：

[复制代码](javascript:void(0);)

1 package com.test;

2 /\*\*

3 \*

4 \* @author 刘玲

5 \*

6 \*/

7 public class TestInteger {

8

9 /\*\*

10 \* @param args

11 \*/

12 public static void main(String[] args) {

13 int i = 128;

14 Integer i2 = 128;

15 Integer i3 = new Integer(128);

16 //Integer会自动拆箱为int，所以为true

17 System.out.println(i == i2);

18 System.out.println(i == i3);

19 System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

20 Integer i5 = 127;//java在编译的时候,被翻译成-> Integer i5 = Integer.valueOf(127);

21 Integer i6 = 127;

22 System.out.println(i5 == i6);//true

23 /\*Integer i5 = 128;

24 Integer i6 = 128;

25 System.out.println(i5 == i6);//false

26 \*/ Integer ii5 = new Integer(127);

27 System.out.println(i5 == ii5); //false

28 Integer i7 = new Integer(128);

29 Integer i8 = new Integer(123);

30 System.out.println(i7 == i8); //false

31 }

32

33 }

[复制代码](javascript:void(0);)

首先，17行和18行输出结果都为true,因为Integer和int比都会自动拆箱（jdk1.5以上）。

22行的结果为true,而25行则为false,很多人都不动为什么。其实java在编译Integer i5 = 127的时候,被翻译成-> Integer i5 = Integer.valueOf(127);所以关键就是看valueOf()函数了。只要看看valueOf()函数的源码就会明白了。JDK源码的valueOf函数式这样的：

1 public static Integer valueOf(int i) {

2 assert IntegerCache.high >= 127;

3 if (i >= IntegerCache.low && i <= IntegerCache.high)

4 return IntegerCache.cache[i + (-IntegerCache.low)];

5 return new Integer(i);

6 }

看一下源码大家都会明白，对于-128到127之间的数，会进行缓存，Integer i5 = 127时，会将127进行缓存，下次再写Integer i6 = 127时，就会直接从缓存中取，就不会new了。所以22行的结果为true,而25行为false。

对于27行和30行，因为对象不一样，所以为false。

我对于以上的情况总结如下：

* 1. 无论如何，Integer与new Integer不会相等。不会经历拆箱过程，i3的引用指向堆，而i4指向专门存放他的内存（常量池），他们的内存地址不一样，所以为false  
       ②两个都是非new出来的Integer，如果数在-128到127之间，则是true,（因为会缓存？）否则为false  
       java在编译Integer i2 = 128的时候,被翻译成-> Integer i2 = Integer.valueOf(128);而valueOf()函数会对-128到127之间的数进行缓存  
       ③两个都是new出来的,都为false  
       ④int和integer(无论new否)比，都为true，因为会把Integer自动拆箱为int再去比

# Gtest

#### 三、布尔值检查

|  |  |  |
| --- | --- | --- |
| **Fatal assertion** | **Nonfatal assertion** | **Verifies** |
| ASSERT\_TRUE(condition); | EXPECT\_TRUE(condition); | condition is true |
| ASSERT\_FALSE(condition); | EXPECT\_FALSE(condition); | condition is false |

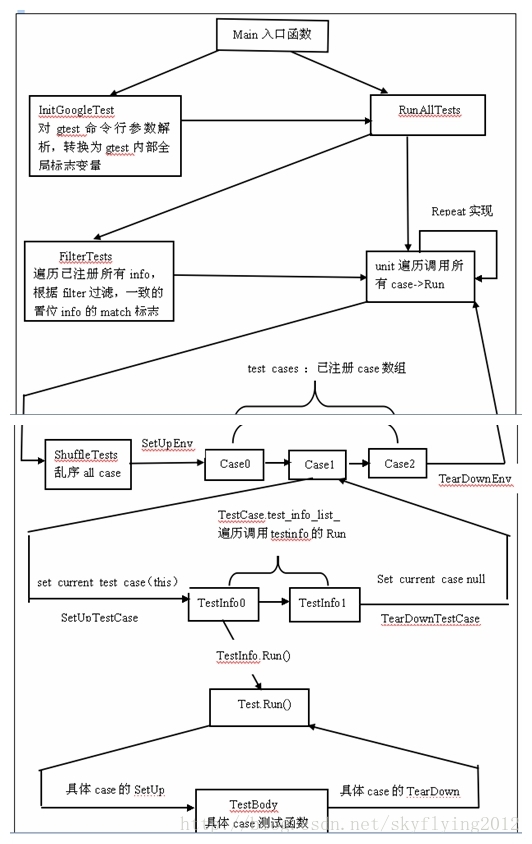
#### 四、数值型数据检查

|  |  |  |
| --- | --- | --- |
| **Fatal assertion** | **Nonfatal assertion** | **Verifies** |
| ASSERT\_EQ(expected, actual); | EXPECT\_EQ(expected, actual); | expected == actual |
| ASSERT\_NE(val1, val2); | EXPECT\_NE(val1, val2); | val1 != val2 |
| ASSERT\_LT(val1, val2); | EXPECT\_LT(val1, val2); | val1 < val2 |
| ASSERT\_LE(val1, val2); | EXPECT\_LE(val1, val2); | val1 <= val2 |
| ASSERT\_GT(val1, val2); | EXPECT\_GT(val1, val2); | val1 > val2 |
| ASSERT\_GE(val1, val2); | EXPECT\_GE(val1, val2); | val1 >= val2 |

#### 五、字符串检查

|  |  |  |
| --- | --- | --- |
| **Fatal assertion** | **Nonfatal assertion** | **Verifies** |
| ASSERT\_STREQ(expected\_str, actual\_str); | EXPECT\_STREQ(expected\_str, actual\_str); | the two C strings have the same content |
| ASSERT\_STRNE(str1, str2); | EXPECT\_STRNE(str1, str2); | the two C strings have different content |
| ASSERT\_STRCASEEQ(expected\_str, actual\_str); | EXPECT\_STRCASEEQ(expected\_str, actual\_str); | the two C strings have the same content, ignoring case |
| ASSERT\_STRCASENE(str1, str2); | EXPECT\_STRCASENE(str1, str2); | the two C strings have different content, ignoring case |

**all case run —— RunAllTests**



# 数据库

Windows下可用mysql和psql二选一

安装的时候按照bubi.jason的

“db” 部分配置

"rational\_db\_type": "psql",

"tmp\_path" : "tmp",

"rational\_string" : "hostaddr=127.0.0.1 port=5432 dbname=bubichain user=postgres password=root",

或者

"rational\_db\_type": "mysql",

"tmp\_path" : "tmp",

"rational\_string" : "hostaddr=127.0.0.1 port=3306 dbname=bubichain user=root password=root",

需要保证：

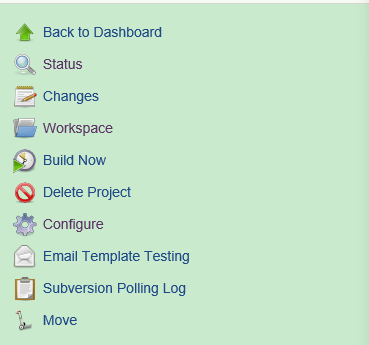
同时只有一个bubi.exe在跑，可以通过任务管理器关掉其他

换数据库时，data下的keyvalue.db文件夹需要删掉重新来

Psql数据库中只留下postgres，bubichain如果是红×就删掉。

# JEKINS

所有文件在work space中，操作脚本shell，makefile等，在config的build中。



# BUBI

server\_ptr\_->addRoute("submitTransaction", std::*bind*(&WebServer::SubmitTransaction, this, std::placeholders::*\_1*, std::placeholders::*\_2*));

将SubmitTransaction函数绑定到 web server

收到SubmitTransaction http请求时，会调用SubmitTransaction接口。

经过一系列错误的排除，调用WebServer::Instance().MakeTransactionHelper(json\_item["transaction\_json"], tran,result)生成和type对应的operation frame 填充ope\_create\_account和acc\_threshold，最后封装成一个protocol::SlaveTransaction slaveTransaction;

将交易信息放在protocol::SlaveTransaction slaveTransaction中。

非slave的时候就bubi::MasterService::GetInstance()->Recv(ZMQ\_NEW\_TX, slaveTransMsg);

放进task\_queue\_.Put(new PipelineTask(this, type, buf.c\_str(), buf.length()));

从taskqueue pop时会调用void PipelineServer::PipelineTask::Run(utils::Thread \*this\_thread)，在run中会调用type(new tx)对应的 handler

void MasterService::OnNewTransaction(const char\* msg, int len, std::*string* &reply)

# vim

:1,$d 删除所有

# gdb

1. 编译时加-g开关。
2. gdb -tui ./cpptest 进入tui
3. tui和非tui的gdb切换：先ctrl+x 再按a
4. 在非tui模式下，l [行号] 将会显示这一行前后5行，一共十行，继续l可以接着显示10行，继续回车（重复上一条命令l）就会一直继续显示10行，直到结束。
5. 打断点，b [行号]
6. 启动，r
7. 下一步（一句一句执行，遇到函数不跳入，直接执行完）n
8. 单步执行，step，不知道s可不可以
9. 回到命令行，ctrl+z，然后可以用vim修改源文件再编译
10. 回到tui，先用job看tui的编号，然后fg [编号]就回去了，并且源代码变成了改过的。

# Tomcat

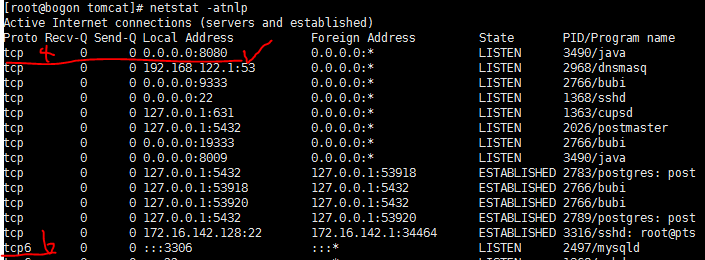
监控中心部署说明

1.安装tomcat 8.0.9 2.安装mysql 创建数据库 monitor ,创建表(导入.sql文件) [root@bogon tmp]#mysql -u root monitor -p < /tmp/monitor\(20161209\).sql 3.修改war包中的配置文件(在WEB-INF里)

global.properties (红色部分为访问的域名)=============== server.url=http\://beta\_monitor.bubidev.cn/ websocket.url=ws\://beta\_monitor.bubidev.cn/nodeInteraction dbconfig.properties (修改红色部分,monitor为数据库名,密码为jar加密后的密码)============== url=jdbc:mysql://localhost:3306/monitor?generateSimpleParameterMetadata=true driverClassName=com.mysql.jdbc.Driver username=root password=JPSg1b5SKhxmiDZlcXDYfvpFQsH7XO0gSNeVqa3Xj/88dhEuvTJg7kyl0qkh5ZBe6d9dZwRvX8c0rysZoQvSdg==

1. bubi\_monitor.war改名为ROOT.war,放到webapps下（tomcat起来后会自动解压为ROOT文件夹，ROOT相当于输入localhost:8080时会跳转的url的所有信息
2. war是一个web模块，其中需要包括WEB-INF，是可以直接运行的WEB模块。而jar一般只是包括一些class文件，在声明了Main\_class之后是可以用java命令运行的。 它们都是压缩的包,拿Tomcat来说,将war文件包放置它的\webapps\目录下，启动Tomcat,这个包可以自动进行解压，也就是你的web目录，相当于发布了。 war包:是做好一个web应用后，通常是网站，打成包部署到容器中
3. ） 5. 修改本机hosts文件 路径: C:\Windows\System32\drivers\etc 末尾追加 192.168.1.154 my\_monitor 6.启动tomcat/bin/.start.sh文件

问题: 1.如果出现jdbc 可能内存泄露提示,查看localhost.log ,之前是因为没有创建表,结果一直提示这个 2.netstat -antlp 查看tomcat8080端口,



是否是ipv4的,如果是ipv6的,需要禁用机器的ipv6

禁用方法：

修改/etc/sysctl.conf中的

net.ipv6.conf.all.disable\_ipv6 = 1

关闭防火墙：

systemctl stop firewalld.service #停止firewall

systemctl disable firewalld.service #禁止firewall开机启动

消除防火墙规则：

iptables -L 列出所有规则  
iptables -F 清除所有

Tomcat 在windows下启动不了怎么办：

http://jingyan.baidu.com/article/ceb9fb10d8bac78cad2ba0db.html

# Internet

域名，输入到浏览器里的网址，DNS解析器会把网址转换成公网ip地址，根据地址找到服务器。公网地址世界唯一。一般服务器的公网地址是固定的。

在测试环境下 只要修改C:\Windows\System32\drivers\etc\host ，把域名和内网ip对应起来就可以用域名访问。

例如加一行

172.16.142.128 beta\_monitor2.bubidev.cn

就可以直接在浏览器里浏览beta\_monitor2.bubidev.cn，前提是172.16.142.128 tomcat是配好且运行着的。

# HTML

**HTML页面加载和解析流程**   
  
1. 用户输入网址（假设是个html页面，并且是第一次访问），浏览器向服务器发出请求，服务器返回html文件。   
  
2. 浏览器开始载入html代码，发现<head>标签内有一个<link>标签引用外部CSS文件。   
  
3. 浏览器又发出CSS文件的请求，服务器返回这个CSS文件。   
  
4. 浏览器继续载入html中<body>部分的代码，并且CSS文件已经拿到手了，可以开始渲染页面了。   
  
5. 浏览器在代码中发现一个<img>标签引用了一张图片，向服务器发出请求。此时浏览器不会等到图片下载完，而是继续渲染后面的代码。   
  
6. 服务器返回图片文件，由于图片占用了一定面积，影响了后面段落的排布，因此浏览器需要回过头来重新渲染这部分代码。   
  
7. 浏览器发现了一个包含一行Javascript代码的<script>标签，赶快运行它。   
  
8. Javascript脚本执行了这条语句，它命令浏览器隐藏掉代码中的某个<style>（style.display=”none”）。杯具啊，突然就少了这么一个元素，浏览器不得不重新渲染这部分代码。   
  
9. 终于等到了</html>的到来，浏览器泪流满面……   
  
10. 等等，还没完，用户点了一下界面中的“换肤”按钮，Javascript让浏览器换了一下＜link＞标签的CSS路径。   
  
11. 浏览器召集了在座的各位<div><span><ul><li>们，“大伙儿收拾收拾行李，咱得重新来过……”，浏览器向服务器请求了新的CSS文件，重新渲染页面。

.com .cn .org 等等 都是域名 和 html无关  
html 是文件格式  
你看到的 xxx.com 其实是 xxx.com/index.html  
index.html 在一些情况下省略了

.html和.jsp的区别

HTML（Hypertext Markup Language）文本标记语言，它是静态页面，和JavaScript一样解释性语言，为什么说是解释性 语言呢？因为，只要你有一个浏览器那么它就可以正常显示出来，而不需要指定的编译工具，只需在TXT文档中写上HTML标记就OK。  
JSP（Java Server Page）看这个意思就知道是Java服务端的页面，所以它是动态的，它是需要经过JDK编译后把内容发给客户端去显 示，我们都知道，Java文件编译后会产生一个class文件，最终执行的就是这个class文件，JSP也一样，它也要编译成class文件！JSP不 止要编译，它还得要转译，首先把JSP转译成一个Servlet文件，然后在编译成class文件。当用户访问JSP时就执行了class文件，最 终......

1.最简单的区别就是，HTML能直接打开，jsp只能发布到Tomact等服务器上才能打开 。

  2.定义上HTML页面是静态页面可以直接运行，JSP页面是动态页它运行时需要转换成servlet。

 3.他们的表头不同，这个是JSP的头“ <%@ page language="java" import="java.util.\*" pageEncoding="gbk"%>”在表头中有编码格式和倒入包等。

 4.也是很好区分的在jsp中用<%%>就可以写Java代码了，而html没有<%%>

# Debug

在什么都对的情况下，转移资产签名错误。

解决：

由于使用redis作为sql的缓存，client key在sql进行了更改，但缓存中还是旧的，执行过程中app接口先去redis里面取了缓存中的key，如果不存在才回去sql里面取，因为redis没清除，所以还是取得redis的。

清除一下redis就好了。

Redis里存的是改变频次低的数据，或者是需要很长时间才能取到的数据。

# J2EE环境搭建

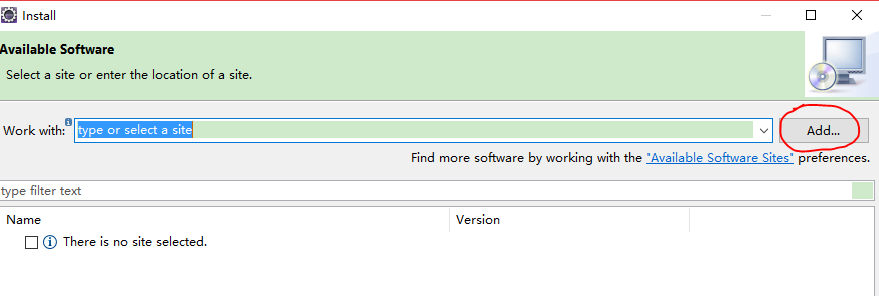
Eclipse：

eclipse-jee-neon-2-win32-x86\_64.zip

插件：

springsource-tool-suite-3.8.3.RELEASE-e4.6.2-updatesite

eclipse打开后help->install new software

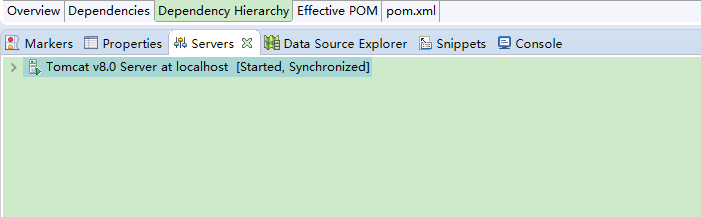


直接打开springsource-tool-suite-3.8.3.RELEASE-e4.6.2-updatesite，解压不解压都可以，然后选择带IDE spring的安装。

项目配置，settings.xml放进.m2里，一般在user/你的名字 路径下面。

Window->show views->others

Server/servers



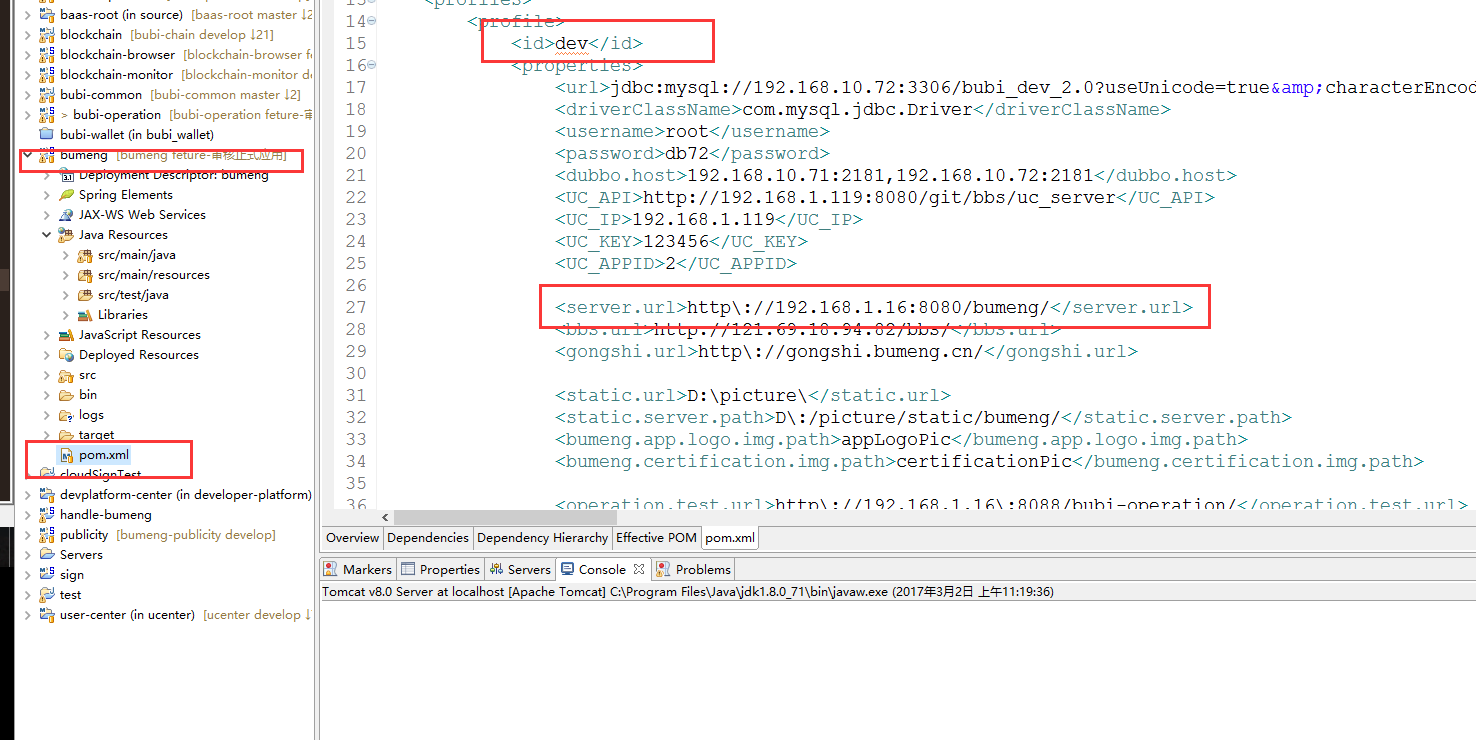
这里右键最上面可以new一个server。从网上下载对应的安装包解压，把路径指向它就可以有server了。

加载项目，右键 maven，找到对应的pom.xml，就打开了

有叉子：

1检查settings.xml对不对

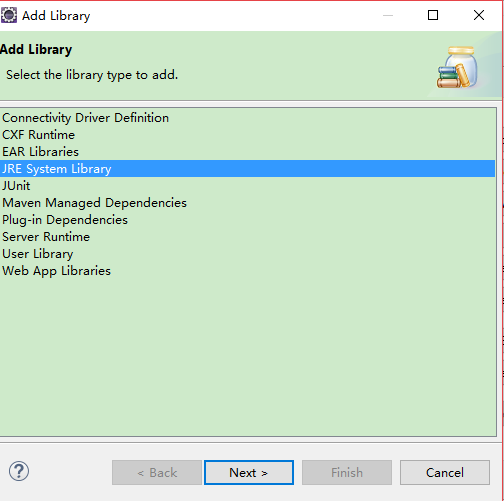
2 项目右键，maven，update project，有时候直接就好了。

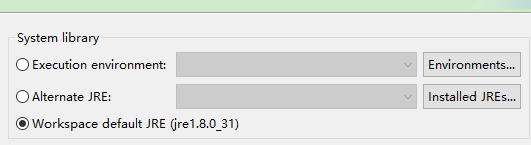


红框里改成localhost:8080/bumeng

生成war包，右键run as ，先maven clean， 再maven install

Install不出来：检查构建路径jre 移除，重新加一遍





# Git

本地的路径中除了一份具体的代码，还有.git目录，内容如下：

每一个项目都只能有一个Git目录，它是为你的项目存储所有历史和元信息的目录，包括上文提到的所有对象(commits、trees、blobs、tags)，而这些对象指向不同的分支。通过指令cd .git查看目录的内容，即可看到所有的重要文件如下：

\* HEAD //表示这个git项目处在哪个分支里

\* config //项目的配置信息，输入`git config`相关命令将会改动它

\* description //项目的描述信息

\* hooks //系统默认钩子脚本目录

\* index //索引文件

\* logs //各个refs的历史信息

\* objects // 本地仓库的所有对象 (`commits`, `trees`, `blobs`, `tags`)

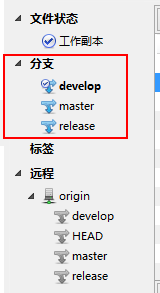
\* refs //项目里的每个分支指向了哪个`commit`

也许通过`cd .git`命令还会看到其他的文件或目录，比如我的项目中还有branches、tree、sourcetreeconfig等，也许你的项目中还有其他与我也不同的东西，不过目前这些不重要，先知道以上几个就好了。

### Git目录与Git工作目录区别

Git的工作目录只是用来临时保存checkout过的用来编辑的文件的地方，可供编辑直到下次commit为止。

这个临时副本属于哪个分支，可以在sourcetree的这个地方切换：

想要切换哪个副本就右键检出哪个，但是并不是从远端checkout，只是在本地的临时版本中切换。

对应git命令git checkout develop



# Python

Window下进入python，powershell打开输入python。

如不识别，执行[Environment]::SetEnvironmentVariable("Path",  
"$env:Path;C:\Python27", "User")

执行脚本：python 脚本名.py

Print语句：print “要print的内容” 或者，print ‘要print的内容’

如使用“”，则里面的‘’不需要转义。如使用‘’，则里面的“”不需要转义。

例如：

print "I'd much rather you 'not'."  
print 'I "said" do not touch this.'

注释符号：#, 在print双引号中写#会直接打印出来,不会被注释掉.

包含日韩中文，需要在开头加入一行：# -\*- coding: utf-8 -\*-

Print+三个” +一段话+三个”可以打印这一段话

Print后面接算式,无引号,直接打印出结果. 结果精确度和运算数据中最精确的一个相同.

两个整数相除,其结果只是整数. 试试 7.0 / 4.0 和 7 / 4 比较一下，你就看出不同了

print "." \* 10 # 打印10个点

%r 打印raw data

print “%r %r %r %r” % (1-2,1+2,3+4,7+8)

获取输入：

age = raw\_input("How old are you? ")

获取函数帮助：

Window下：python -m pydoc raw\_input

其他：pydoc raw\_input

使用参数：

from sys import argv  
script, first, second, third = argv  
print "The script is called:", script  
print "Your first variable is:", first  
print "Your second variable is:", second  
print "Your third variable is:", third

调用时：python ex13.py first 2nd 3rd

其中 ex13.py是script，其他是输入参数。

打开文件函数：

txt = open(filename)

如果只写 open(filename) 那就使用 'r' 模式打开的吗？  
是的，这是 open() 函数的默认工作方式。

打印文档内容

print txt.read()

txt = open(filename) 返回的是文件的内容吗？  
不是，它返回的是一个叫做“file object”的东西

• close – 关闭文件。跟你编辑器的 文件->保存.. 一个意思。  
• read – 读取文件内容。你可以把结果赋给一个变量。  
• readline – 读取文本文件中的一行。  
• truncate – 清空文件，请小心使用该命令。  
• write(stuff) – 将 stuff 写入文件

target = open(filename, 'w')

除了 'w' 以外，我们还有 'r' 表示读取（read）， 'a' 表示追加(append)。

target.write(line1)  
target.write("\n")  
target.write(line2)  
target.write("\n")  
target.write(line3)  
target.write("\n")

Note  
注意到我在每行 print 后面加了个逗号(comma) , 了吧？这样的话 print 就不会输出新行符而结束  
这一行跑到下一行去了。

len(indata)#读取长度byte

exists(to\_file) #判断文件存在，to\_file是file object

由于Python是使用缩进来标识代码块，因此，不像[C/C++](https://www.baidu.com/s?wd=C%2FC%2B%2B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YzPWDkuWc4rHwWnyfLnjTL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHb4P1b3PWfsPHmLrjDYPWfd)这类语言，必须有{}  
在python中，任何具有相同缩进量的代码会被识别为一个代码块，当下一行语句与上一行的缩进量不同时，它就自动退出了上一行语句所属的代码块

def print\_two(\*args):

arg1, arg2 = args

print "arg1: %r, arg2: %r" % (arg1, arg2)

函数名称有什么规则？  
和变量名一样，只要以字母数字以及下划线组成，而且不是数字开始，就可以了。

f.seek(0)#指针回到开头

f.readline()#读取第一行

f.readline()#读取第二行

每次你运行 f.seek(0) 你就回到了文件的开始，而运行  
f.readline() 则会读取文件的一行，然后将“磁头”移动到 \n 后面。

readline() 里边的代码会扫描文件的每一个字节，直到找到一个 \n 为止，然后它停止读取文件，并且返回此前的文件内容。文件 f 会记录每次调用 readline() 后的读取位置，这样它就可以在下次被调用时读取接下来的一行了。

words = stuff.split(' ')  
return words

return sorted(words)

这种写法是按照0~9，a-z 排序

如果是数组

>>> students = [('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10),]

用key函数排序：**返回由tuple组成的list**

**Python代码**

1. >>> sorted(students, key=lambda student : student[2])   # sort by age
2. [('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]

其中student叫啥都行  
用cmp函数排序

**Python代码**

1. >>> sorted(students, cmp=lambda x,y : cmp(x[2], y[2])) # sort by age
2. [('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]

**用 operator 函数来加快速度,**

**Python代码**

1. >>> from operator import itemgetter, attrgetter
2. >>> sorted(students, key=itemgetter(2))

用 operator 函数进行多级排序

**Python代码**

1. >>> sorted(students, key=itemgetter(1,2))  # sort by grade then by age
2. [('john', 'A', 15), ('dave', 'B', 10), ('jane', 'B', 12)]

If语句

if cars > people:

print "We should take the cars."  
elif cars < people:

print "We should not take the cars."  
else:

print "We can't decide."

For循环：

the\_count = [1, 2, 3, 4, 5]

fruits = ['apples', 'oranges', 'pears', 'apricots']

change = [1, 'pennies', 2, 'dimes', 3, 'quarters']

for number in the\_count:

print "This is count %d" % number

for fruit in fruits:

print "A fruit of type: %s" % fruit

for i in change:

print "I got %r" % i

#用什么表示数组里的一个元素都可以，只要数组名对就行。

elements = []  
for i in range(0, 6):

print "Adding %d to the list." % i

elements.append(i)

while循环

i = 0  
numbers = []  
while i < 6:  
print "At the top i is %d" % i  
numbers.append(i)  
i = i + 1  
print "Numbers now: ", numbers  
print "At the bottom i is %d" % i

simple-todo笔记：

1. 教程地址：<http://simple-is-better.com/news/309>
2. 导入.sql的方法，1.安装mysql.2.用mysql cmdline client创建名为todo的database，再导入.sql（导入命令**D:\APMServ5.2.6\MySQL5.1\bin>mysqldump -u root -p  erp lightinthebox\_tags > ligh  
   tinthebox.sql**）没成功，直接进到todo的database里，然后把.sql里面的东西直接复制出来粘到cmdline里就可以了。
3. 要安装web.py,网上下载，然后解压放到应用的文件根目录,到powershell里python setup.py install
4. 要安装mysqldb，网上找一个装。
5. 如果templates报错，就把setting.py里的render路径改为绝对路径。
6. 运行：可以用notepad++运行code.py,但是会造成static路径下的东西找不到（没有样式）。需要用powershell。
7. 默认static下放所有静态文件，但是不知为何notepad++不行。

# 正则表达式

\w: \w 与以下任意字符匹配：A-Z、a-z、0-9 和下划线 和汉字

\b:\b匹配这样的位置：它的前一个字符 和后一个字符不全是(一个是,一个不是或不存在)\w

如果要精确地查找hi这个单词的 话，我们应该使用\bhi\b

假如你要找的是hi后面不远处跟着一个Lucy，你应该用\bhi\b.\*\bLucy\b。

这里，.是另一个元字符，匹配除了换行符以 外的任意字符。\*同样是元字符，不过它代表的不是字符，也不是位置，而是数量——它 指定\*前边的内容可以连续重复使用任意次以使整个表达式得到匹配。因此，.\*连在一起就意味着任意数量的不包含换行的字符。现在\bhi\b.\*\bLucy\b的意思就很明显了：先是一个单词 hi,然后是任意个任意字符(但不能是换行)，最后是Lucy这个单词。

0\d\d-\d\d\d\d\d\d\d\d匹配这样的字符串：以0开头，然后是两个数字，然后是一个连字号“-”，最后是8个数字(也就是中国的电话号码。当然，这个例子只 能匹配区号为3位的情形)。

这里的\d是个新的元字符，匹配一位数字 (0，或1，或2，或……)。-不是元字符，只匹配它本身——连字符(或者减号，或 者中横线，或者随你怎么称呼它)。

为了避免那么多烦人的重复，我们也可以这样写这个表达式：0\d{2}-\d{8}。 这里\d后面的{2}({8})的意思是前面\d必须连续重复匹配2次(8次)。

\s匹配任意的空白 符，包括空格，制表符(Tab)，换行符，中文全角空格等。

然后是任意数量的字母或数字(\w\*)

\d+匹配1个或更多连续的数字。 这里的+是和\*类似的元字符，不同的是\*匹配重复任意次(可能是0次)，而+则匹配重复1次或更多次。

| 表1.常用的元字符 | |
| --- | --- |
| **代码** | **说明** |
| . | 匹配除换行符以外的任意字符 |
| \w | 匹配字母或数字或下划线或汉字 |
| \s | 匹配任意的空白符 |
| \d | 匹配数字 |
| \b | 匹配单词的开始或结束 |
| ^ | 匹配字符串的开始 |
| $ | 匹配字符串的结束 |