# 开发日志

## 数据采集工具在执行任务报错：

###### 1.1、权限不够：给对应的数据库赋权限，在对应的模式下打开命令行执行：

grant all on schema 模式名 to 用户名；

grant all on all tables in shcema 模式名 to 用户名;

###### 1.2、XXX 视图或表 not exists ：在中间库或融合库下的dpksh模式下缺少依赖的表或视图：

重新导入表或视图

###### 1.3、执行关于金钱任务的跑数任务时报错：integer of out range：

将该任务对应的表的对应金钱字段从 int4 修改为 numeric 类型，由于plsql没有numeric 类型，修改为decimal 类型 ，保存后会变为numeric 类型。

注意：需要将hybzk\_dpksh和dpzjk下的所有涉及到该字段的int类型修改

1. 1.3.1、在修改的过程中可能会遇到报错内容为依赖视图存在无法修改：

将所有的依赖视图的语句先拷贝下来并删除，修改后重新执行视图语句

1. 1.3.2、需要将gp库下的hyrhk模式和中间库的dpzjk模式下的使用到的表全部修改

###### 1.4 执行任务出错，空间不足

执行任务的所在的硬盘满了，导致的原因可能是因为没有升级数据采集工具

任务不会自动删除执行的任务，解决方法，删除所在路径冗余的数据

## 在使用指标计算平台的报错

###### 2.1、新建指标报错：执行sql语句错误：

需要将sql语句放到gp库下的ajzj模式下执行一遍，如果有错误则修改；如果没有错误则检查关联表是否都有”模式名”.”表名”，若没有则需要添加

###### 2.2、导入任务无法初始化：

1、在将任务部署到现场的过程中需要将任务导出、并且导入到现场的指标计算平台中，在修改数据源之后无法初始化任务：平台的bug，不能将修改数据源，需要将导入的数据源的ip地址修改为现场的ip地址即可。

2、将任务中涉及到的指标和任务重建，重新初始化和移送

3、将任务删除重建，并且修改表名

###### 2.3、现场在修改计算任务和指标的时候会将关联的表和视图删除

###### 2.4、由于指标计算平台的问题导致任务表和视图无法创建或删除

1、首先向开发反映问题

2、可以选择手动创建或删除表和视图

###### 2.5java.io.FileNotFoundException:t\_jspt\_qd\_sy\_ajsj/gatheredToMonth/data/db\_dpzjk.t\_jspt\_qd\_sy\_ajsj\_mouth.dat (No such file or directory)

解决方法：导入指标计算平台任务的文件中创建一个data文件夹gatheredToMonth/data（gatheredToDay/data）

###### 2.5查看指表计算平台的配置

1、查看指标计算平台的服务器地址（pwd:6789@jkl）🡪进入隐藏目录（cd /data）🡪进入TAS8080文件夹 🡪查看jvm.ini(cat jvm.ini)🡪获取服务器配置地址🡪使用管理员用户名和密码登录（username: zbjspt; pwd:123 ）

2、向指标计算平台的开发人员进行求助

###### 2.6 现场数据采集工具报错503

解决方案：找到对应的安装包，重启Tas服务

## python Script

###### 3.1、写脚本的时候需要考虑到数据为None(null)的情况 和‘’的情况

###### 3.2、在给标签置0或‘’之后 判断 是否为‘’ 就可以

###### 3.3、清除缓存脚本

setDataDigits("","总数");#从有数据切换到无数据的时候会保留上一次的数据。通过这个来清空数据缓存-陈真凤

setDataDigits( "","新收");

setDataDigits( "","旧存");

setDataDigits("" ,"已结");

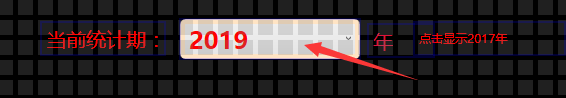
setDataDigits("" ,"未结");

###### 3.4、使用python脚本获取当前时间

## 设计器使用

###### 1、根据年限选择统计期，并展示对应年份的数据

1.1、用到了新的控件“选择框”(下拉选项框)



1.2选项值采用key-value的选项存储的，可以设置默认项索引（输入序号从0开始）

1.3赋值：

1.3.1 如果选项不会变化可以直接输入选项的key 和value

1.3.2 选项会变化可以采用绑定数据集的方式获取选项

拟定可以选择近三年，故使用数据集赋值；数据集不需要查询表直接进行结果查询即可

数据集sql： select (EXTRACT(year from now())-2)::varchar as y ,'0' as i

union all

select (EXTRACT(year from now())-1)::varchar as y ,'1' as i

union all

select (EXTRACT(year from now()))::varchar as y ,'2' as i

1.4获取选项值：

绑定在选择框控件 选择项改变

Python脚本：

from VisualDiagram.Event import \*

from VisualDiagram.Utility import \*

from System.Windows.Media import \*

def getSelectedValue():

#获取当前选择的值

value=this['xlk'].GetProperty("SelectedName");

#弹出当前选择的值

MessageBox.Show(value);

def SelectDemo()：

#给选择框选择序号为0的选项

   this["选择框1"].SelectedIndex=0

1.5 sql传参查询

demoSQL:

select

columnvalue5 as year,

columnvalue11 as lx,

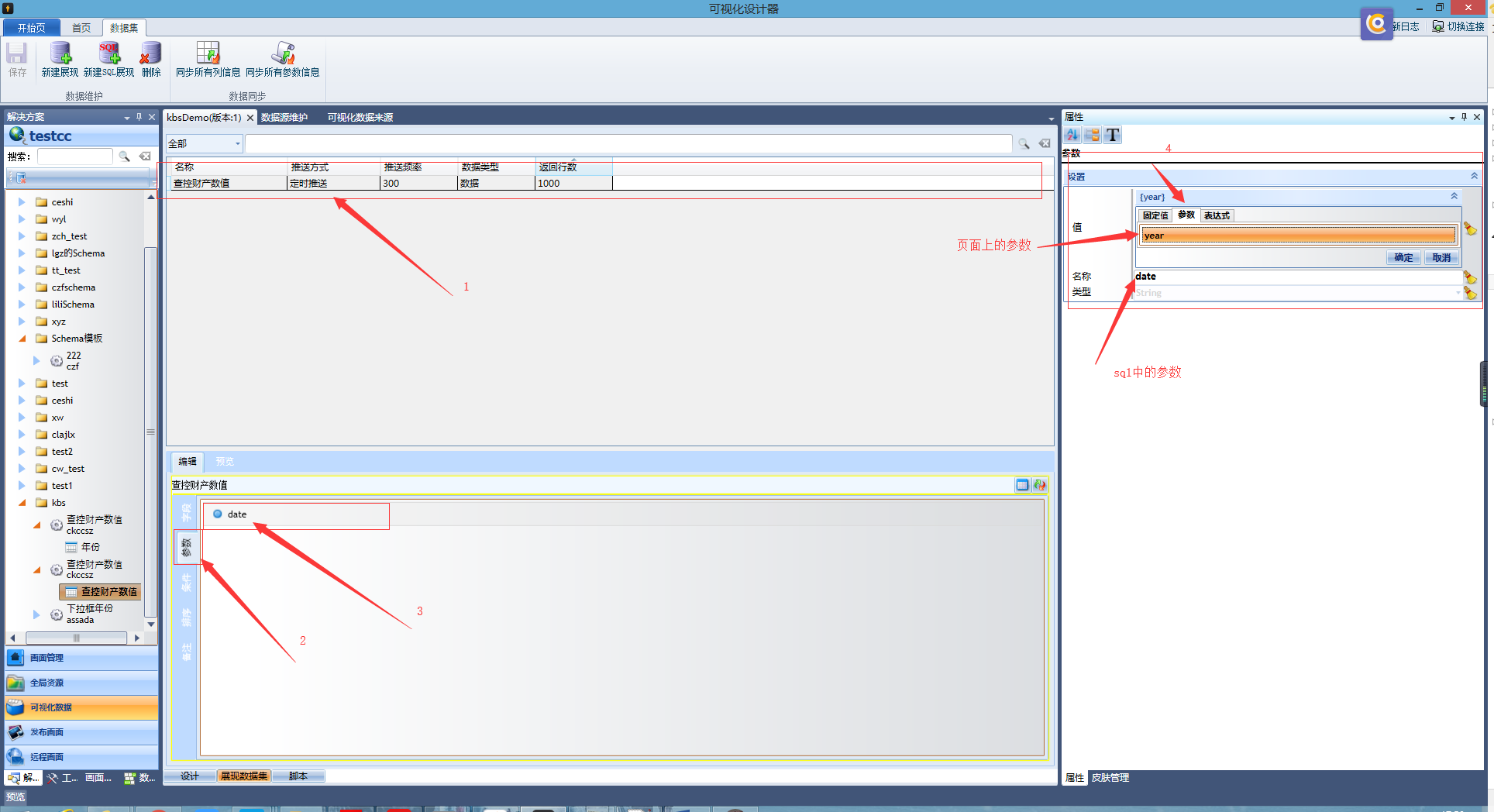
columnvalue12 as value

from db\_cc.z\_visu\_kbs

where schemaid='ckccsz' and columnvalue5='${date}'

参数date作为查询条件，这样一次性读取出来的数据量少，设计器不会卡顿

1.6将带参数的数据集绑定在控件上面，并将页面参数传递到sql的参数中



将sql中有参数的数据集放到展现数据集中🡪选中参数🡪显出sql中定义的参数

🡪选中右侧会显示出设置栏🡪设置参数，将页面参数和sql中的参数进行对应。

🡪在给页面参数赋值后数值也会随之改变

1.7在柱图刷新数据的时候，遇到了数据库中没有的年份，则会遗留上次选择有年份的数据，即没有年份无法刷新。故某年没有数据的正确表现形式为数据库数据库中存在某年 但是数据列为null。

###### 2. 页面上数据在等待5分钟之后会产生改变的情况

原因：在数据集刷新数据的时候会初始化参数，或者读取其他预览同一画面的参数

## 部署数据库

###### 1、还原数据库时报错：

重新打包一个包发过来然后重新还原成功

###### 2、备份postgresql数据库：

在本地的\thunisoft\arterybase\3.5\bin目录下Shift+鼠标右击🡪选择在此处打开命令行窗口🡪输入命令：pg\_dump -h localhost -U username databaseName > filename.bak

命令含义： -h:要备份的数据库所在的服务器地址（localhost）

-U:用户名(username)

要备份的数据库名称（databasename）

文件名称(filename.bak)

🡪回车后输入密码，不报错即可成功

###### 3、还原postgresql数据库：

详情大屏现场部署文档

###### 4、安装postgresql中无法通过ip访问数据库

1、修改pg\_hba.conf 中的配置 ：

# TYPE DATABASE USER CIDR-ADDRESS METHOD

host all all 0.0.0.0/0 md5

或者增加

host all all 0.0.0.0/0 true

2、修改postgresql.conf

listen\_addresses = '\*'

## 开启服务遇到问题

###### 1、查看license是否过期

使用的是正版过期找恒泰重新申请新的license

使用的是临时版本（只有一天期限），删除licence文件及缓存重新启动

###### 2、webcontext实例化失败（河南高院）

查看tomcat启动的进程 (命令： ps -ef|grep tomcat )

是否有重复启动的tomcat，如果有杀掉所有的该tomcat的进程（命令: kill -s 9 id）

然后重新启动

###### 3、webcontext实例化失败（洛阳中院）

问题排查过程：

1、liscence查看有没有过期，没有过期

2、 查看是否有重复启动的tomcat：没有

查看方法：在Xshell输入命令：ps -ef|grep tomcat 会列出所有正在运行的tomcat的进程

3、查看是端口是否被占用：没有

查看方法：在Xshell输入命令： netstat -tunlp 会列出正在使用的端口

4、怀疑tomcat和visu的包出现问题，从217服务器上下载下来山东高院的包，重新修改配置，启动失败，山东高院的包是可以正常启动的

5、将修改之后的里面的数据库地址改成山东高院的地址，启动成功，改成洛阳的数据库地址，启动失败。得出结论：洛阳的数据库出现了问题

6、重新恢复洛阳的数据库

7、在本地启动成功

8、去服务器启动原来tomcat，启动成功。问题解决

9、大屏无法访问可视化数据，修改数据源。问题解决

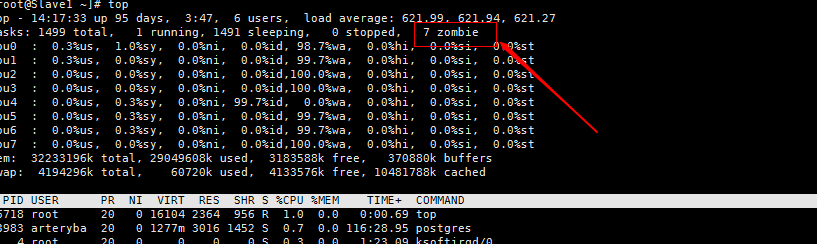
###### 4、服务器无法启动大屏服务和备份数据库

1、场景：大屏服务无法启动，试了几次，输入命令之后会卡死。

数据库无法备份。

2、查看进程（:ps -ef|grep tomcat）显示有进程。

3、查看内存和进程使用情况(:top)



发现僵尸进程和孤儿进程（defunct）

4、猜想可能是因为这个原因造成服务器不能正常使用

使用各种方法杀掉僵尸进程和孤儿进程，无效

僵尸进程需要寻找到它的父进程的id，才能kill掉。在此处无效。

5、孤儿进程无法kill掉只能重新启动服务器

6、打算重启服务器，输入各种重启和关机命令均无效，会卡在命令执行的界面

7、使用虚拟机控制重启，经过寻找虚拟机的登录帐号和密码错误

8、经过百度查询强制重启方法后重启成功。

9、成功解决

10、启动arterybase，需要切换到到普通用户，查找所有用户，切换到arterybase用户，命令:su username。服务启动成功，大屏服务启动成功，数据库备份成功，服务器恢复正常

###### 5、关于断电启动虚拟机环境

登录到VmwareESXI

217地址：172.23.20.213 username/pwd: root/123qwe!@#

114地址：172.23.18.100 username/pwd : root/6789@jkl

指标管理系统：172.16.32.220 /data/tas5080

起220服务器找北京的人（陶珊珊）--》起arterybase服务—》起TAS服务

指标计算平台数据库：172.16.22.37

配置路径<http://172.16.11.60/appconfig（zbjspt/123>）

6、webcontext实例化失败（青岛现场）

问题：服务启动不了，因为jdk版本不对应

解决方法：切换jdk版本

## sql 调优

1、EXISTS与IN的使用效率的问题，通常情况下采用exists要比in效率高，因为IN不走索引，但要看实际情况具体使用：

IN适合于外表大而内表小的情况；EXISTS适合于外表小而内表大的情况。

2、应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描

3、应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描

4、不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

5、事务语法：原子性（Atomicity，或称不可分割性）、一致性（Consistency）、隔离性（Isolation，又称独立性）、持久性（Durability）。

## 网络问题/软件问题

###### 1、关于teamViewer13连接超时的解决办法

a.使用电脑管家或360卸载teamViewer13软件

b.修改MAC地址（network address）设备管理器🡪网卡🡪属性🡪网络地址修改为12个字符的地址

注意：一定要记住本机的mac地址，在命令行中 ipconfig -all 的物理地址为mac地址

c.修改完mac地址后，重新安装teamViewer13.

d.安装完成之后，将mac地址修改为原来的地址

00-23-24-E8-F6-EE

###### 2、同一局域网内，一台机器可以访问sso（单点登录）服务器，另一台则不能

解决方法：修改网络的DNS解析服务器地址

## 常用liunx命令

1、cat XXX：查看某个文件内容

2、cd XXX:移动到某个目录下

3、vi XXX：使用vi编辑器编辑某个文件

4、netstat -tunlp:查看占用的端口

5、sudo：以系统管理者的身份执行指令

6、echo’message’ > test.txt :向文件中写入信息

7、ps -ef 显示正在运行的所有的进程 ps -ef|grep tomcat:筛选tomcat的进程

8、查看所有用户 cat etc/passwd

9、切换用户 su username

10、按照名称查找对应的文件

find / -name \*name\*

find (path) -name 正则匹配

11、./Starup.sh(运行.sh的文件)

12、将文件（目录）展示出来

ll：展示详细信息

ls：展示名称

13、、查看liunx硬盘空间

14、赋权限：chmod 777 filename[正则匹配文件(\*.sh)]

15、tail -f

tail 用于查看文件的内容 -f参数：动态查看文件内容（从最后显示，并且动态刷新）

## Git学习日志

公司Git：username/pwd:mengxinwb/6789@jkl

1.git的三种状态

已提交（committed）--已经保存到本地的数据库

已修改(modified) --文件已经修改未存到数据库

已暂存(staged)—表示对一个已经修改的文件做了标记，使之包含在下次提交的快照之中

2.git的三个工作区域

Git仓库(git resource)

用户的的工作目录(user resource)

暂存区域(stage )

3.git的基本工作流程

在用户的工作目录中修改文件🡪暂存文件，将修改后的文件快照放入暂存区🡪提交更新，找到暂存区的文件快照，将快照存储到git仓库中去

4.git的安装及配置

下载地址：<https://git-scm.com/download/win/>安装方法百度

安装成功之后进行配置

Windows：运行🡪cmd🡪git config –list显示出配置列表

配置用户信息（在以后的每次提交中git都会使用配置的这个用户信息）：

$git config --global user.name “username”

$git config --global user.email [email@qq.com](mailto:email@qq.com)

$git config <key> ：查看某项配置信息

5.1git基础-获取git仓库

关键英文:仓库(repository)、跟踪(track)、暂存(stage)、提交(commit)、推送(push)、拉取(pull)

获取git项目仓库的两种方法：

在现有的项目或者目录下导入所有的文件到git中

如果你打算使用git来对现有的项目进行管理，你只需要进入该目录并输入

$git init

此命令将创建一个名为.git的子目录，这个子目录含有你初始化的git仓库中所有的必须的文件这些文件是git仓库的骨干。但是项目中的文件还没有被跟踪。

$git add \*.txt

$git add LICENSE

$git commit -m ‘initial project version’

通过git add 命令来实现对指定文件的跟踪(将文件放入暂存区) 和使用commit 来提交

从一个服务器克隆一个现有的git仓库

克隆的仓库的命令git clone [url]，假如要克隆git的可连接库libgit2，可以使用

$ git clone <https://github.com/libgit2/libgit2> [自定义文件夹名称]

这会在当前目录下创建一个名为libgit2的目录，并在这个目录下初始化一个.git文件夹，从远程的仓库拉取下来所有的数据放入.git文件夹中，然后从中读取最新版本的文件进行拷贝

5.2git基础-记录每次更新到仓库

1. 检查当前文件状态：git status

git status -s

$ git status -s

M README

MM Rakefile

A lib/git.rb

M lib/simplegit.rb

?? LICENSE.txt

A：代表是新增加到暂存区的文件

M:代表是已经修改过的文件，左边的M代表是该文件被修改并放入了暂存区

1. 忽略文件可以创建一个.gitignore的文件，列出要忽略的文件模式

cat .gitignore

\*.[oa]

\*~

第二行告诉git忽略所有以 .a或.o结尾的文件。

第三行告诉git 忽略以~结尾的文件

git diff 来分析文件差异

1. 提交更新

当暂存区域已经准备妥当可以提交了。在此之前，一定要确认还有什么修改过的或新建的文件还没有git add过，否则不会记录这些还没有暂存的变化。所以每次准备提交前，先用git status看一下，是不是都已经暂存起来了，然后再运行提交命令：git commit  
git commit -m”message”:提交暂存区的文件并且记录message

git commit -a -m“message”跳过暂存区直接将文件保存到git存储库中去

1. 移除文件

Git rm –cached fileName ：git不再跟踪这个文件

git rm -f fileName :强制删除已经add后的文件，并且不再跟踪

1. 查看提交历史

git log 查看所有的提交历史，最近的提交在最上面显示

git log -p -2 :查看最近两次提交的差异

1. 撤销操作

假如在提交完了才发现漏掉了几个文件没有添加，或者提交信息写错了

git add newFile

git commit --amend -m“message”

执行完这个代码之后上次的提交会被覆盖掉，提交的信息也会被修改为第二次的信息

1. 取消暂存的文件

假如使用了git add \* 增加了所有的文件，其中有一些是不想暂存的则可以使用下面的命令来取消这些文件的暂存

git reset HEAD <fileName>

5.3git基础-远程仓库的使用

查看已经配置的远程仓库服务器命令：git remote -v (需要在指定的地址)

查看远程仓库的详细信息 使用命令:git remote [remote-name] [branch-name]

添加一个新的远程仓库：git remote add <shortname> <url> （shortname可以代替url）

如果想要拉取某个仓库中有但自己没有的信息可以使用代码：

git fecth shortname

如果使用git clone 命令克隆了一个git远程仓库，会自动将其添加到远程仓库并默认以origin为简写，所以git fetch origin 会抓取上一次后新推送的所有的工作，但是不会自动合并到自己工作区中，需要手动合并。

推送到远程仓库

git push [remote-name][branch-name]

只有当你有服务器的写入权限，并且之前没有人推送过时，这条命令才能生效。当你和其他人在同一时间克隆，他们先推送到上游然后你再推送到上游，你的推送就会毫无疑问的被拒绝。你必须先将他们推送的新的信息拉取下来合并到你的工作后才能推送

远程仓库的移除与重命名

远程仓库的重命名：git remote rename oldName newName;

注：这样也会修改你远程分支的名字，过去引用的oldName/master 现在的会引用paul/master

远程仓库的移除：git remote rm [remote-name]

5.4git基础-打标签

和其他的版本控制系统（vcs）一样，git可以给历史中的某一个提交打上标签，以示重要。比较具有代表性的是人们会使用这个功能来标记发布节点

列出标签：git tag

是按照字母的顺序列出标签；但是他们出现的顺序并不重要

创建标签

git的标签主要使用两种类型：轻量标签(lightweight)和附注标签(annotated)

创建一个标签，并且给标签添加注释信息

git tag -a v1.4 -m’message’

6.Git分支-分支简介

创建分支：git branch testing

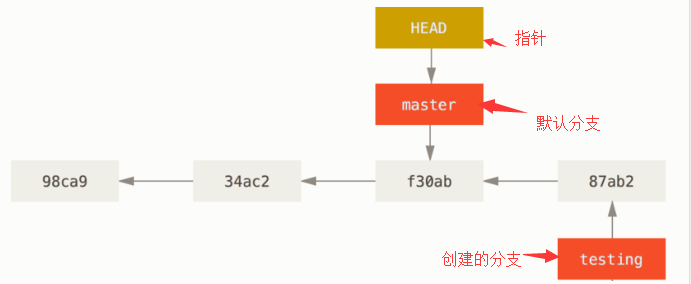
这会在当前所在的提交对象上创建一个指针

git branch 命令仅仅创建一个新的分支，并不会自动切换到新的分支上去

查看各个分支当前所指的对象。命令：git log -decorate

分支切换：git checkout testing

这样HEAD就指向testing分支了



注：当切换到testing的分支下进行开发并且提交后记录会向前进行推进，当用户再次切换到master的分支下时，内容仍是f0ab的内容，综上，从不同的分支可以进行不同的开发流程

1. 删除分支：git branch -d testing 将testing分支删除
2. 合并分支：git merge -testing 将testing分支合并到主分支下

6.1git分支-分支的新建与合并

应用场景：

1. 开发某个程序
2. 正常开发实现新的需求创建一个分支iss53
3. 在iss53上展开工作
4. 此时有一个紧急的bug需要修补，操作步骤如下
5. 切换到你的线上分支
6. 为这个紧急的任务新建一个分支，并在其中修复它
7. 在测试完成之后，切换到线上分支，然后合并这个修补的分支，最后将改动推送到线上分支，推送完成之后删除修复分支
8. 切换回iss53的开发分支，继续开发

git综合

1. 笔记一：git commit 、git push 、git pull、git fetch、git merge的含义与区别

1、git commit :是将本地修改过的文件提交到本地库中

2、git push:是将本地仓库的最新信息发送给远程仓库；

3、git pull:是从远程仓库获取最新版本到本地，并自动merge

4、git fetch:是从远程仓库获取到最新版本到本地，不会自动merge；

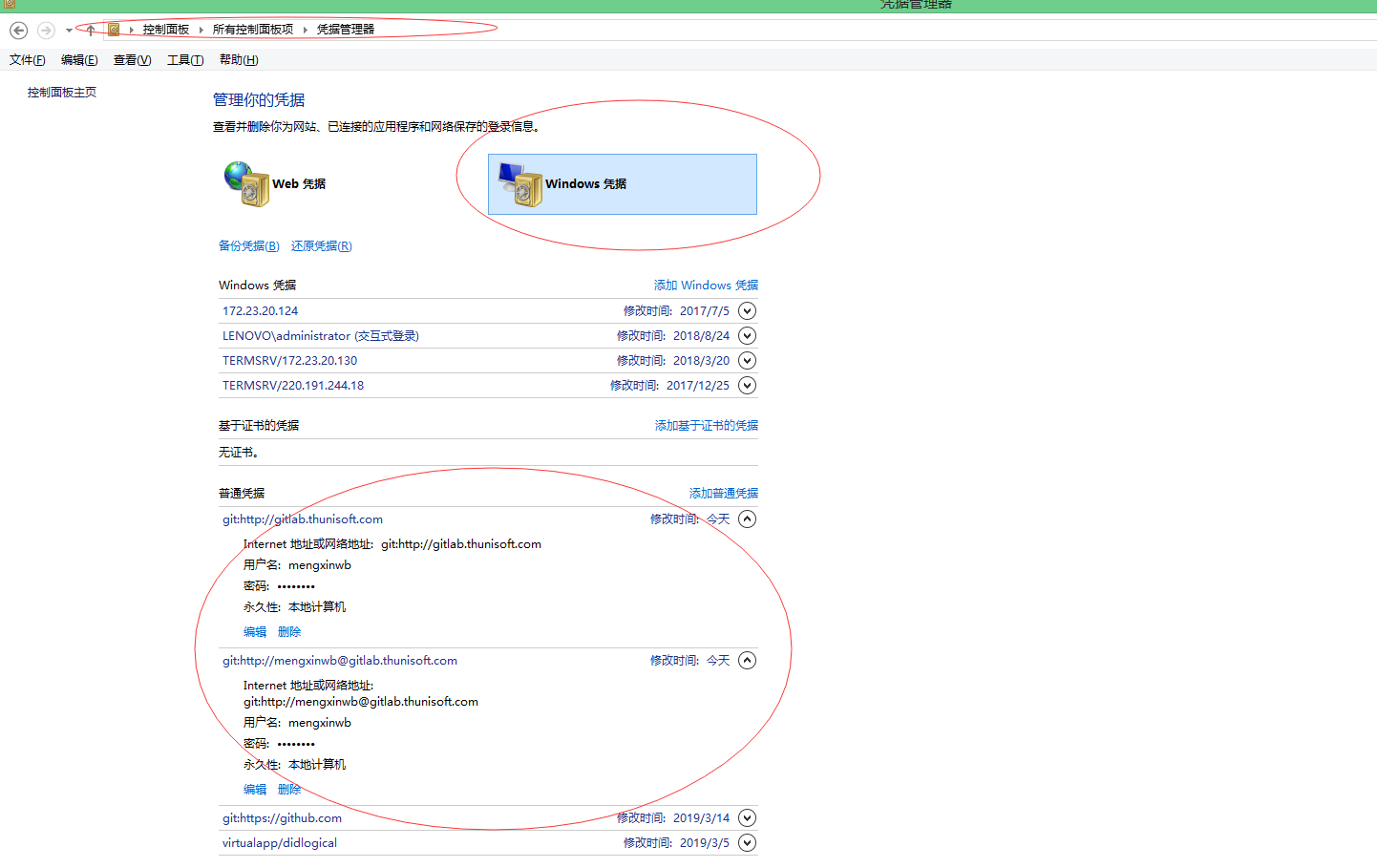
5、git merge:是用于从指定的commit合并到当前分支，用来合并两个分支 $ git merge -b//将b分支合并到当前分支，有时候会产生合并冲突（同时修改了一个文件的相同部分），需要手动修改，修改完成之后我们可以用git add 来告诉git冲突已经解决，并且重新提交

git pull =git fatch+git merge

6、git commit -am’message’：跳过add的步骤直接提交,

git切换用户的时候注意事项

1. 导入一个新的项目的时候clone一个库报错：



如图所示，查看用户名和密码是否正确，若新创建的项目则需要增加这两个账户（准确的说是三个，还有一个SSH）

## IntellJ IDEA 的使用

### 1、创建web项目

##### 创建JavaProject项目

开始页面🡪Cretae New Project 🡪Java🡪next(选择是否创建helloworld方法)🡪next

🡪输入ProjectName、Project location🡪Finsh

##### 创建Module

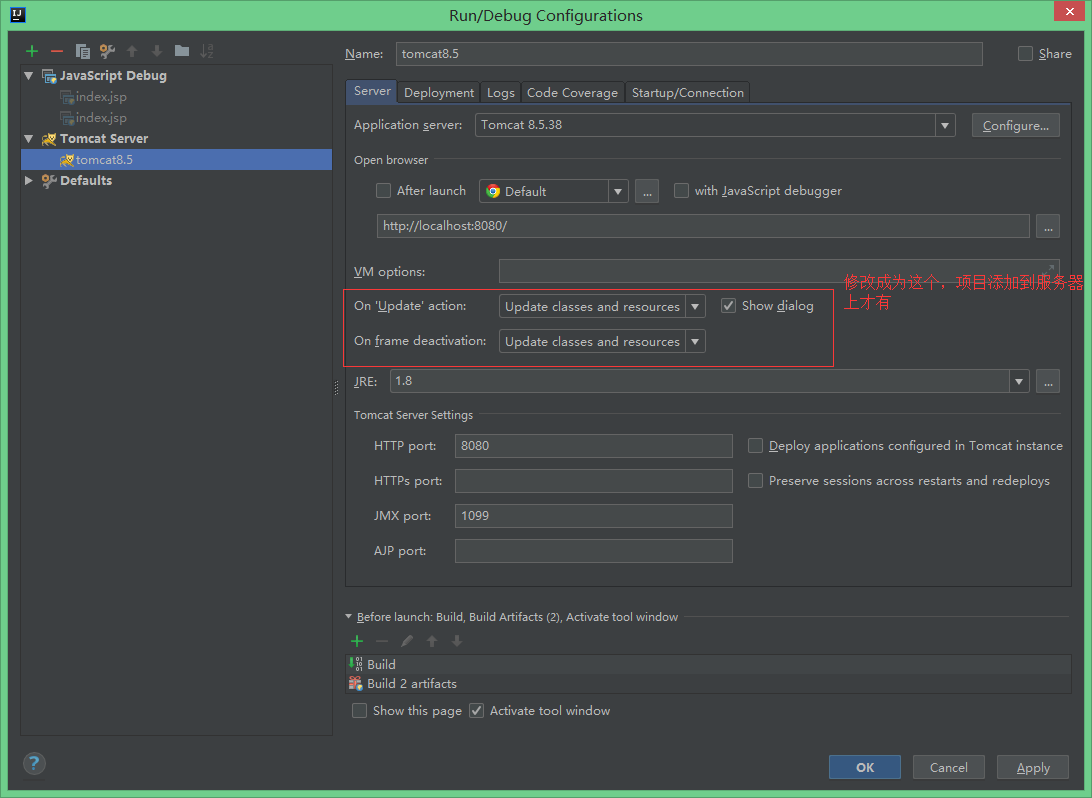
选择项目名🡪右击New🡪Module🡪选择需要创建的类型🡪输入Module name\根路径\路径

### 2、在IDEA中增加热部署（JRebel插件）

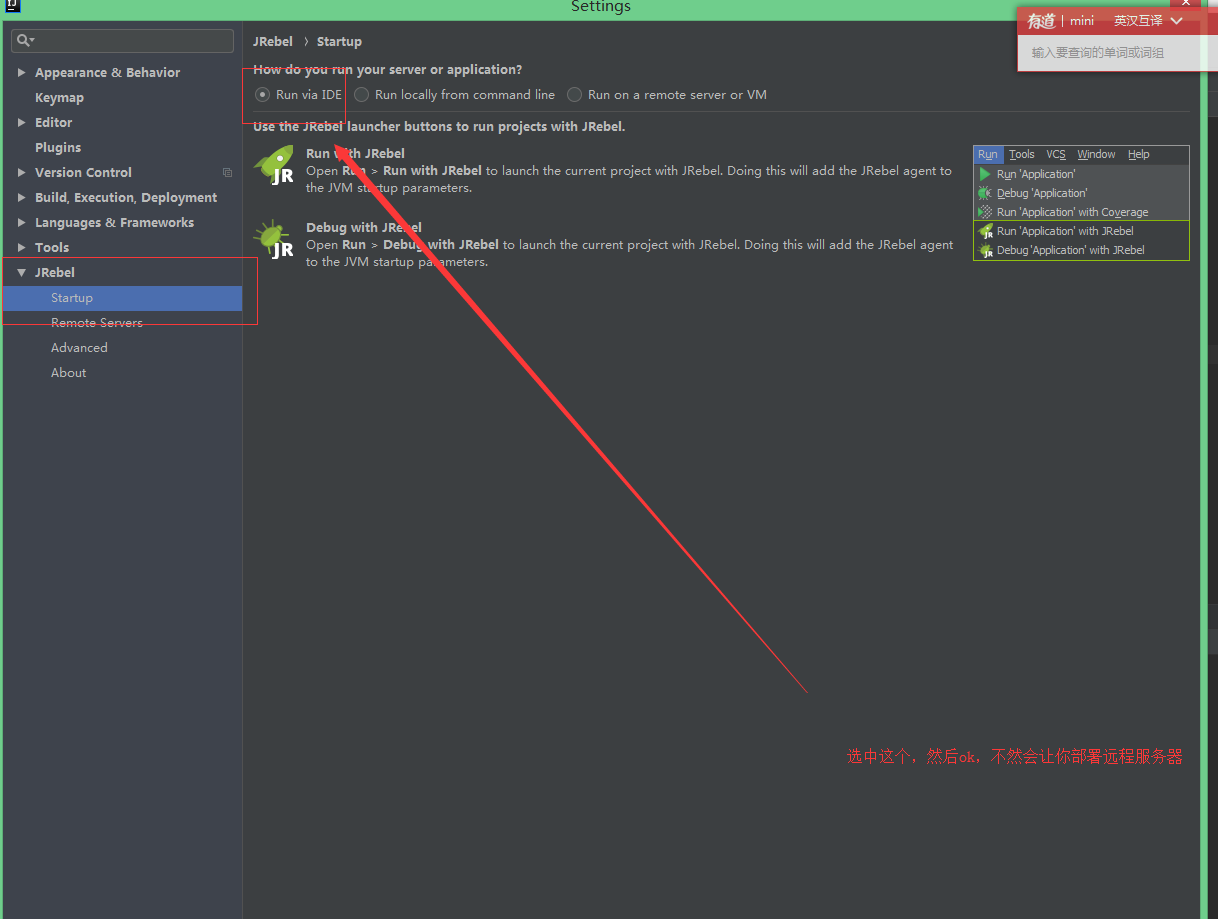
file🡪settings🡪Plugins🡪Browse repositries🡪搜索Jrebel🡪安装🡪重启IDEA

1. 设置启动的tomcat

Edit Configuriations

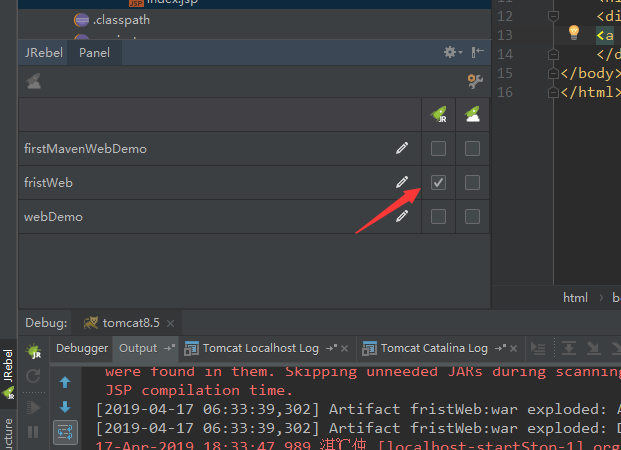


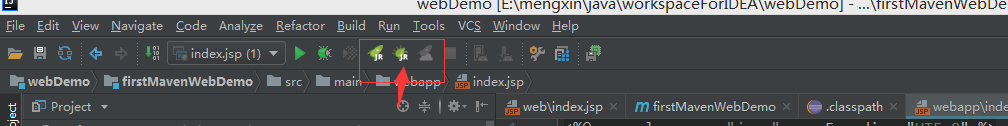
1. Jrebel设置



1. 显示出Jrebel的视图

选中需要热部署的项目

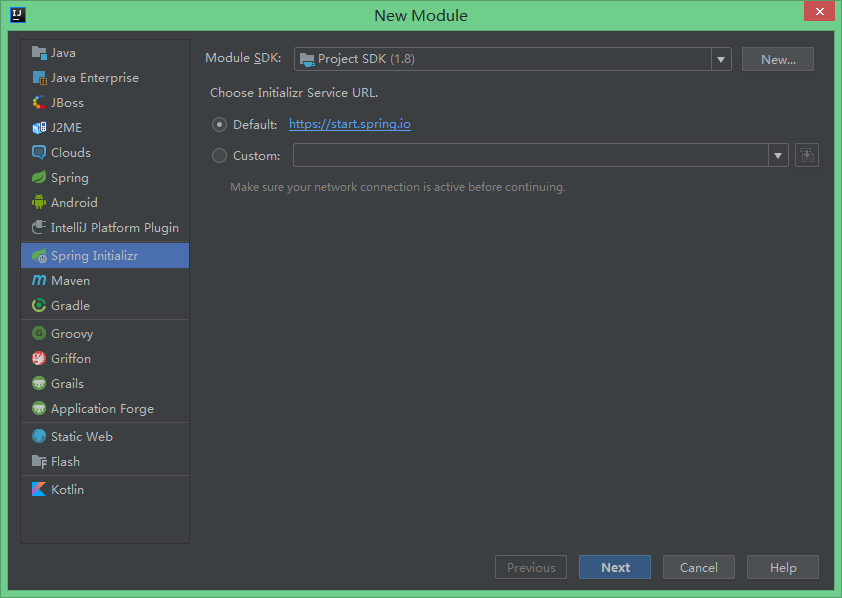




启动热部署

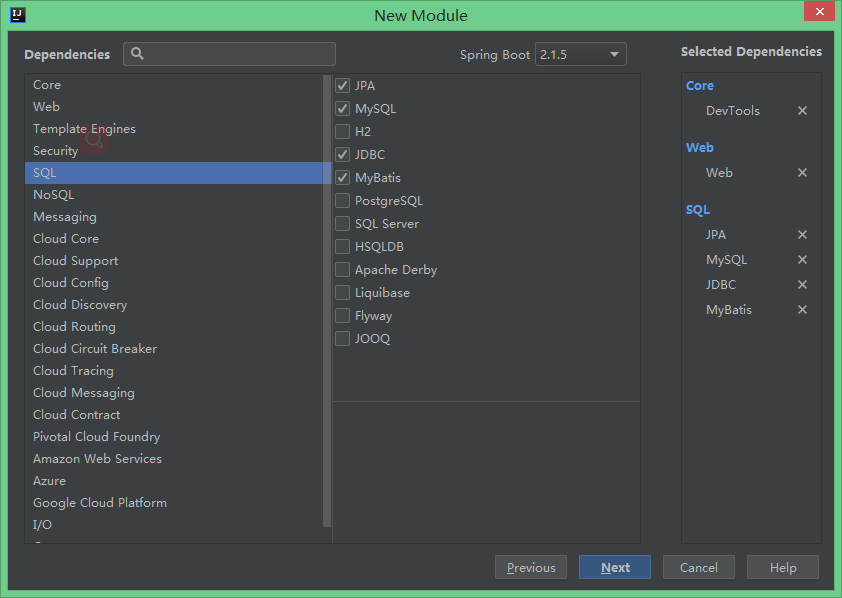
### 创建springboot项目集成mybatis 和log4j

#### 创建springboot项目



#### 填写module的名称和信息

#### 选择对应的组件



#### 如果需要有新增的组件，在网上搜索pom配置，添加即可

##### 添加log4j的配置

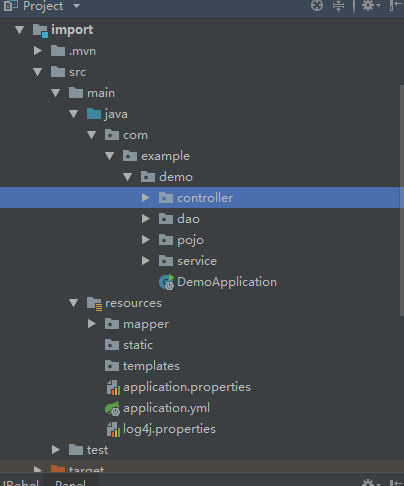
1. 先在application.properties中增加

mybatis.configuration.log-impl = org.apache.ibatis.logging.stdout.StdOutImpl

2、在log4j中增加

#打印sql部分  
log4j.logger.java.sql=DEBUG  
log4j.logger.java.sql.Connection = DEBUG   
log4j.logger.java.sql.Statement = DEBUG   
log4j.logger.java.sql.PreparedStatement = DEBUG   
log4j.logger.java.sql.ResultSet = DEBUG  
  
#配置logger扫描的包路径 这样才会打印sql  
log4j.logger.com.example.demo.dao=DEBUG

#### 创建项目工程常用目录



#### 问题1：IDEA springboot中无法读取.properties文件

解决办法：

setting🡪Editor🡪File Types

选择Properties 添加\*.properties

将Text中的.properties删除

#### 问题2：添加application.properties文件为springboot的启动文件

解决办法：

Project Structure(单个项目/module设置)

Facets🡪spring🡪点击“+”号🡪New Application Context🡪点击“+”号🡪add properties File 🡪选择文件

#### 配置mybatis

需要注意的点：

1. Springboot 启动类中增加@MapperScan(“需要扫描的mapper接口所在的包”)
2. 将mapper.xml文件移动至resources目录下，创建对应的包名
3. 在application.propreties文件中增加配置

#mybatis 配置扫描mapper,将xml文件放在resources目录下

mybatis.mapper-locations = classpath:mapper/\*Mapper.xml

#配置扫描实体类

mybatis.type-aliases-package = com.example.demo.pojo

1. 在dao接口上增加@Repository注解标注该接口为dao

在mapper接口上增加@Mapper注解该接口为mapper

#### maven Dependencies报红

将pom.xml中的dependencies节点剪掉重新进行 Reimport all

导入完成后粘贴dependencies节点后进行Reimport all

## Maven学习日志

###### Maven概述（java project）

要用java开发一个系统，需要涉及到很多的模块，web应用服务器、文件服务器、db等。我们要开发这些模块，需要先把他们各自需要依赖的jar包或者项目下载打包好，然后配置到classpath中。这些应用服务器在运行单元测试pr编译或者部署的时候，需要依赖本地的一些配置，比如jdk、web容器、依赖jar包，这样我们将项目分享出去的时候，别人要使用就有一定的配置门槛。

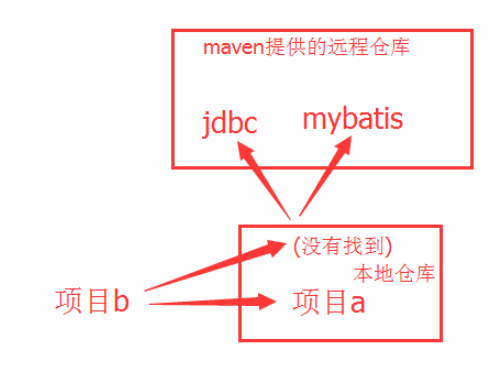
maven可以帮我们完成上述所有的工作，不再需要手动去完成。

maven将项目看成一个对象。

maven建立了一个仓库体系，来管理这些项目。

maven的仓库分为本地仓库和远程仓库（maven提供的中央仓库）

比如我们可以将项目a通过maven的命令放在本地仓库中去，如果项目b需要依赖项目a，则可以在本地仓库中找到。如果需要依赖一些别人的项目，比如jdbc和mybatis，maven会先根据我们配置的项目属性去本地仓库中找，若找不到就去中央仓库中去找。



maven项目对象部分基础属性：

1. 、项目坐标：

项目坐标用来定位一个项目在仓库中的位置。当项目b依赖项目a或者jdbc的时候，maven会根据项目坐标属性去从仓库体系中寻找项目，并自动配置到项目依赖中。

1. 、打包方式：

这个属性决定我们执行maven的package方法后，项目会被打包成什么文件

1. 、项目依赖列表：

这个属性的值是一个数组，里面有一些其他项目的坐标

1. 、构建插件列表。

这个属性是一个数组，定义项目需要的一些插件，比如源码打包工具、web容器

maven项目对象的部分方法：

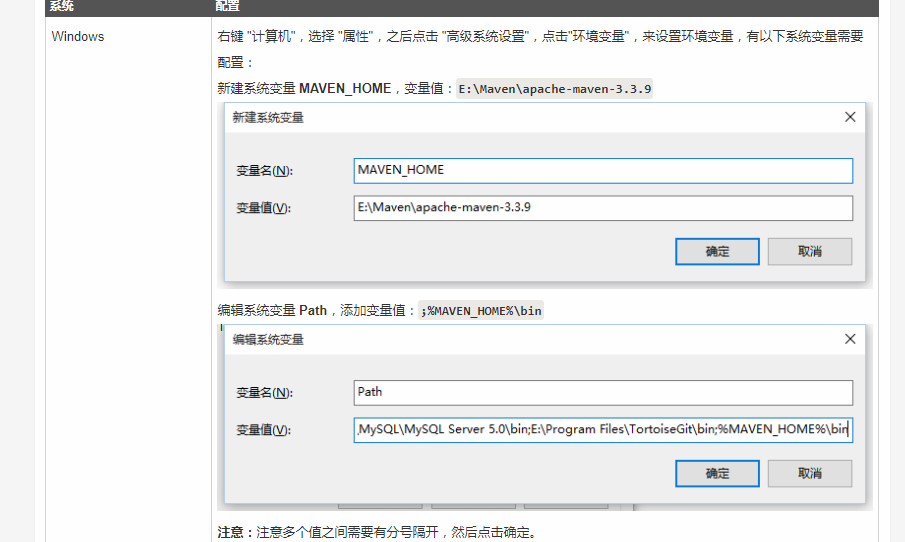
1. mvn compile：编译项目
2. mvn package：打包项目
3. mvn clean：清除之前编译和打包产生的文件
4. mvn install：将项目暗转到本地仓库
5. mvn test 运行测试代码

###### 1、maven的安装及环境配置

下载：百度自行下载，下载完成之后解压到对应的目录（全英文路径）

设置Maven的环境变量

如图所示：



###### 2、Maven POM(Project Object Model,项目对象模型)

1、pom 是maven工程工作的基本单元，是一个XML文件，包含了项目的基本信息，用于描述项目如何构建，声明项目依赖。

当执行任务或者目标时，maven会在当前目录中查找POM。他读取POM，获取所需要的配置信息，然后执行目标

2、POM中指定的配置：

项目依赖、插件、执行目标、项目构建profile、项目版本、项目开发者信息、相关邮件列表信息。



3、所有的pom文件都需要有project元素和三个必需的字段：groupId，artifactId，version。

project：工程的根标签

mode|Version:模型版本需要设置为4.0？？？？

groupId：工程组的表示。他在一个组织或者项目中通常是唯一的

artifactId：这是工程的标识，它通常是工程的名称

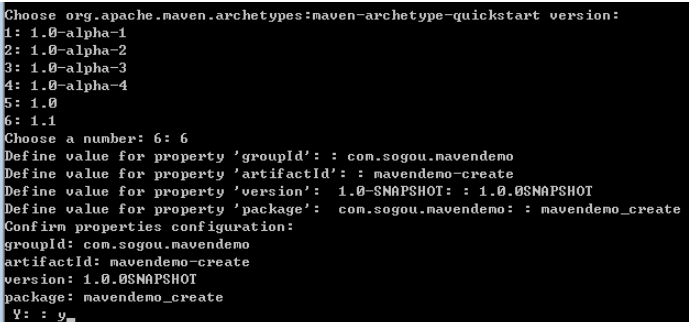
version：工程的版本号。

###### 3.使用maven创建一个java项目（IntelliJ IDEA）

1. 打开cmd，切换到工作目录，运行：mvn archetype:generate

首次运行这个命令会下载很多的依赖文件，不用理会

1. 填写quickstart参数



输入y确认参数后回车，这样就创建了一个maven项目。

我们填写的几个参数：

groupId:是帮助项对象在仓库中定位的

artifactId：决定了生成的项目文件夹目录的名称

version：项目版本号

package：是项目source文件下最顶层的包名



## 大屏数据采集工具部署

## 安徽监察委项目（java）

###### 框架artery4.0()

###### 开发环境：

1. eclipse
2. jdk1.8
3. tomcat8.5
4. maven
5. git

###### 提供git帐号和权限，clone项目源码

mengxinwb 6789@jkl

[git切换用户的时候注意事项](#_git切换用户的时候注意事项)

###### 导入maven项目

配置对应的maven：

环境变量M\_2

在eclipse中配置maven的settings.xml

等待maven导入成功

## 监察委-廉政档案平台项目

### 开发环境和工具:

jdk1.8、tomcat9 、eclipse、maven、postgresql、navicat、Artery5

### 问题记录：

1. 注解没有加@AutoWried
2. 查询数据库名称没有加模式名

Artery表单区域实现可编辑 在跳转的时候将FromType 修改为5 将表单下面的控件的展现类型修改为inherit

1. 分页不可用：数据源没有使用如下写法

return context.queryPageDataForMybatis(qp, new IMybatisQueryWrapper() {

@Override

public Collection executeQuery() {

return tLzdaSslpljjtdjqkMapper.selectByRyid(ArteryParamUtil.getString("ryid"));

}

});

## JAVA报错日志及解决方法

### SSM框架：Could not open ServletContext resource [/db.properties]

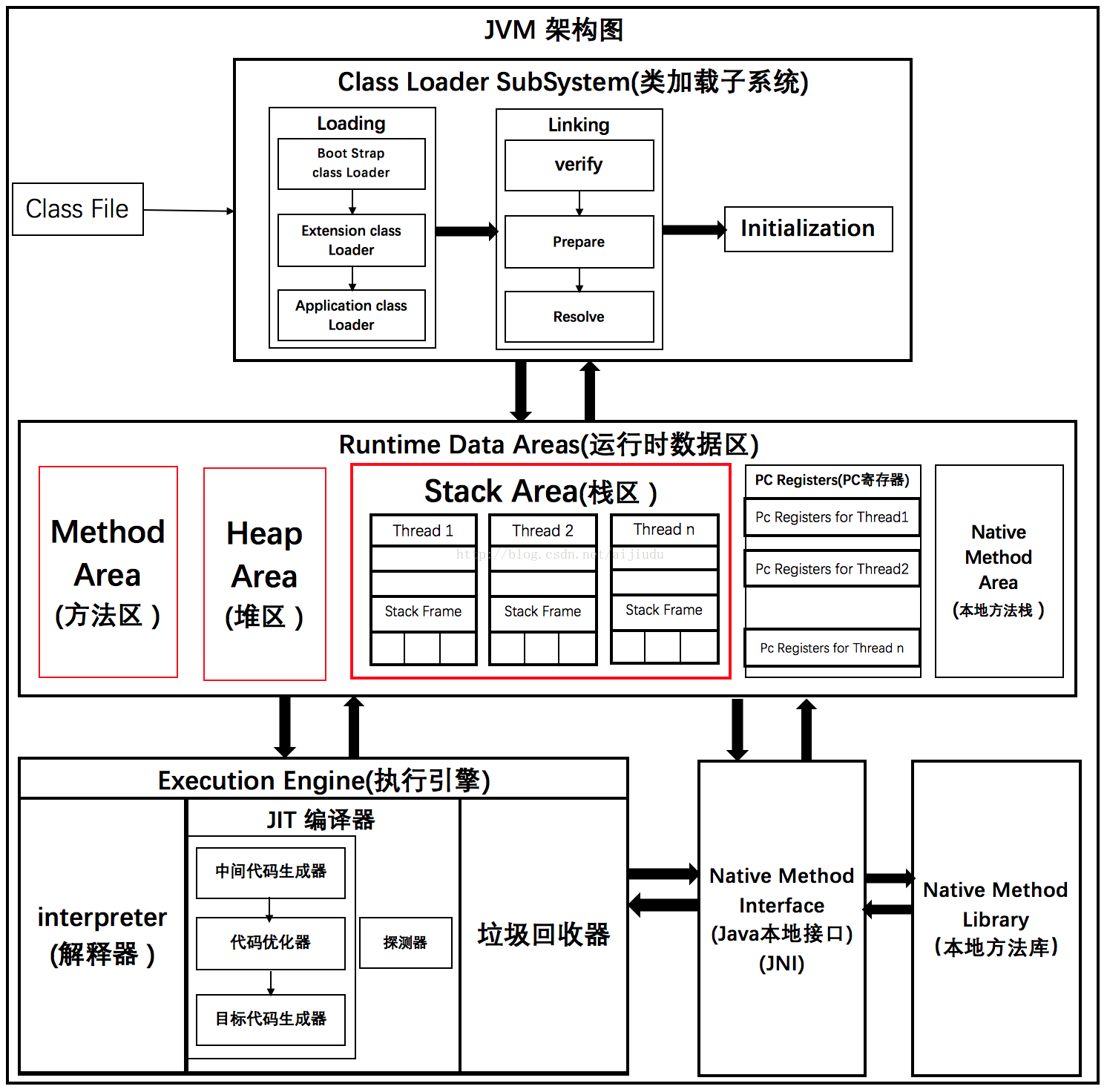
解决办法：在Spring-MVC.xml配置文件中读取.properties文件配置上加上classpath

<context:property-placeholder location="classpath:db.properties" />

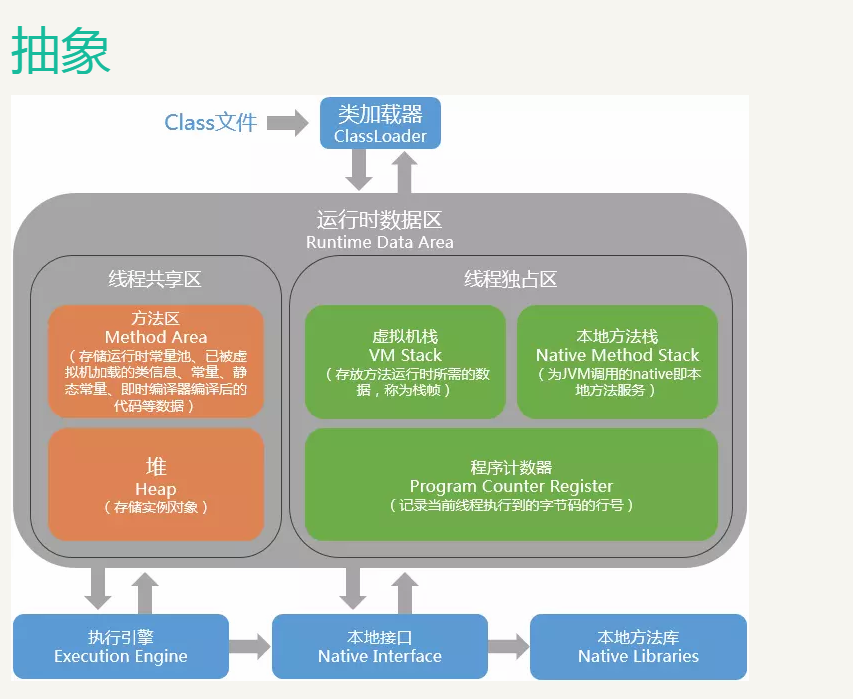
## JAVA学习

#### JVM逻辑内存模型，运行时的内存分配（java运行时内存分配）

###### JVM架构图

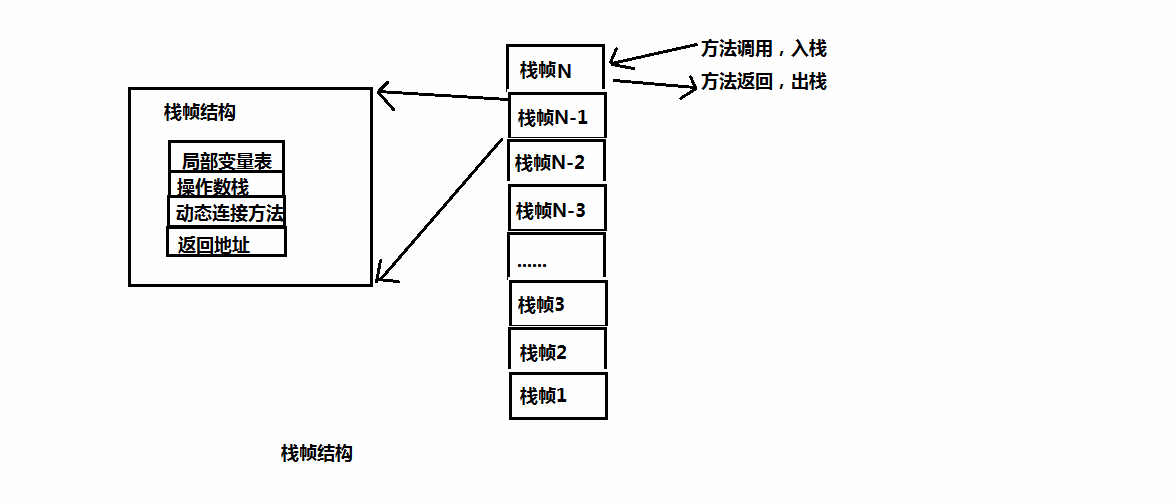


###### Java虚拟机在执行Java程序的过程中，会把它管理的内存划分为几个不同的数据区域，这些区域都有各自的用途、创建时间、销毁时间



分析：

1. 程序计数器：
   1. 用来保存当前正在执行的程序的内存地址，由于java是多线程的，所以程序执行的轨迹不可能是线性执行的，有多个线程交叉执行时，被中断的线程程序当前执行到哪条内存地址必然要保存下来，以便于被中断的线程恢复执行时再按照被中断的指令继续执行下去。为了线程之间切换能够准确，每个线程都需要有一个独立的程序计数器，各个线程之间互不影响，独立储存，我们称为这类内存区域为线程私有的内存，是线程安全的。
2. java栈： java Stack 线程私有的区域。故存放的是基本类型变量、对象或方法的引用



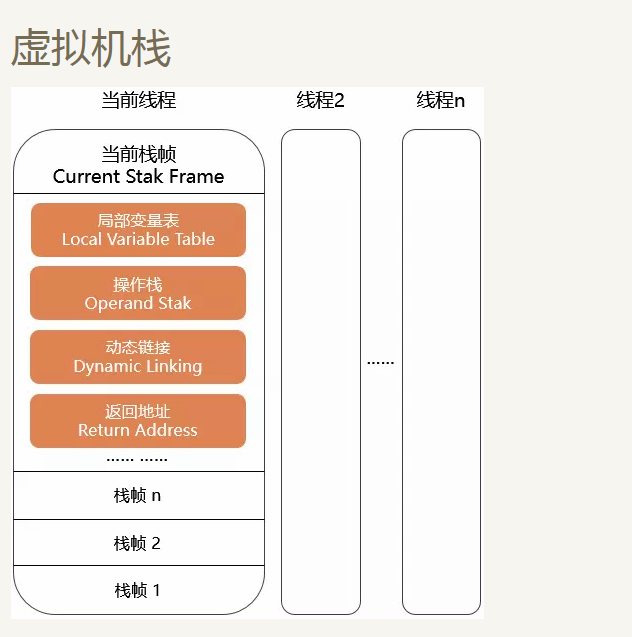
java的栈是和线程关联在一起的，每创建一个线程JVM就为该线程创建对应的java栈，Java栈中包含多个栈帧，这些栈帧是和每个方法关联起来的，每运行一个方法就创建一个栈帧，每个栈帧会有一些局部变量、操作栈和方法返回值等信息。当每一个方法执行完毕时，该栈就会弹出栈帧元素作为这个方法的返回值，并且清除这个栈帧，java栈的栈顶是当前正在执行的活动栈帧，也就是当前正在执行的方法，程序计数器会指向该地址，此时只有当前栈帧中的变量可以被操作栈使用，当该栈帧在调用另外一个方法时，与之对应的一个新栈帧会被创建，这个新创建的栈帧被放到java栈的顶端，变成当前的活动栈，当当前的方法执行完毕时，该栈帧会被移除java栈，之前的栈帧变为当前的活动栈，前面的栈帧的返回值变为当前活动栈帧的一个操作数。

由于java栈是和线程对应起来的，java栈中的数据只为当前线程所用，所以不需要关心该数据的一致性，也不存在同步锁的问题（synchrinized）

在java虚拟机规范中，对该区域规定了两种异常状况，：

当线程请求的栈深度大于虚拟机所允许的深度，将抛出StackOverFlow异常；

如果虚拟机可以动态扩展，若扩展的过程中无法请求到足够的内存，就会抛出OutOfmemoryError异常



3、堆Heap：线程共享的区域。故存放的是对象、方法等信息

堆是jvm中管理内存中最大的一块，是被java线程锁共享的，不是线程安全的，在jvm启动时创建。堆是存储java对象的地方

1. 方法区Method Area：

方法区存放了要加载的类的信息、类中的静态常量、类中的final的常量、类中的方法信息，当程序中通过Class对象的getName.isInterface等方法来获取信息时，这些的数据来源于方法区。方法区是被java线程锁共享的。方法区也是堆中的一部分，就是我们所说的java堆中的永久区，它的大小可以通过参数设置。它不会被GC频繁的回收，存储的信息相对稳定



1. 运行时常量池Constant Pool：

常量池本身是方法区中的一个数据结构。常量池中存储了如字符串final变量值，类名个方法名常量。常量池在编译期间就被确定，并保存在已编译的.class文件中。一般分为两类：字面量和应用量

字面量：就是字符串、final变量等。类名和方法名属于引用量

1. 本地方法栈Native Method Stack：

本地方法栈和java栈的作用类似，区别是java栈是为JVM执行的方法服务，而本地方法栈为JVM执行Native方法服务。

Native方法:一个native方法就是一个java调用非java代码的接口，代表该方法的实现由非java语言实现比如用c/c++实现。

在定义一个native方法的时候，并不提供实现体（类似定一个 java Interface），因为实现体是由非java语言在外面实现的。

扩展：native关键字修饰的方法是一个原生态的方法，方法对应的实现不是在当前的文件，而是在用其他语言实现的文件中。Java语言本省不能对操作系统底层进行访问，但是可以通过JNI（JNI： java native Interface java本机接口）接口调用其他语言来实现对底层的访问

#### JVM的垃圾回收机制

简介：对于程序计数器、虚拟机栈、本地方法栈这三个部分而言，其生命周期是于线程有关，随着线程而生，随着线程而灭，并且这三个区域的内存分配与回收具有确定性，因此当线程结束的时候，内存就自然跟着线程回收了。

因此JVM的内存分配和内存回收关注的是Java堆与方法这两个区域。

##### 如何判断对象已“死”

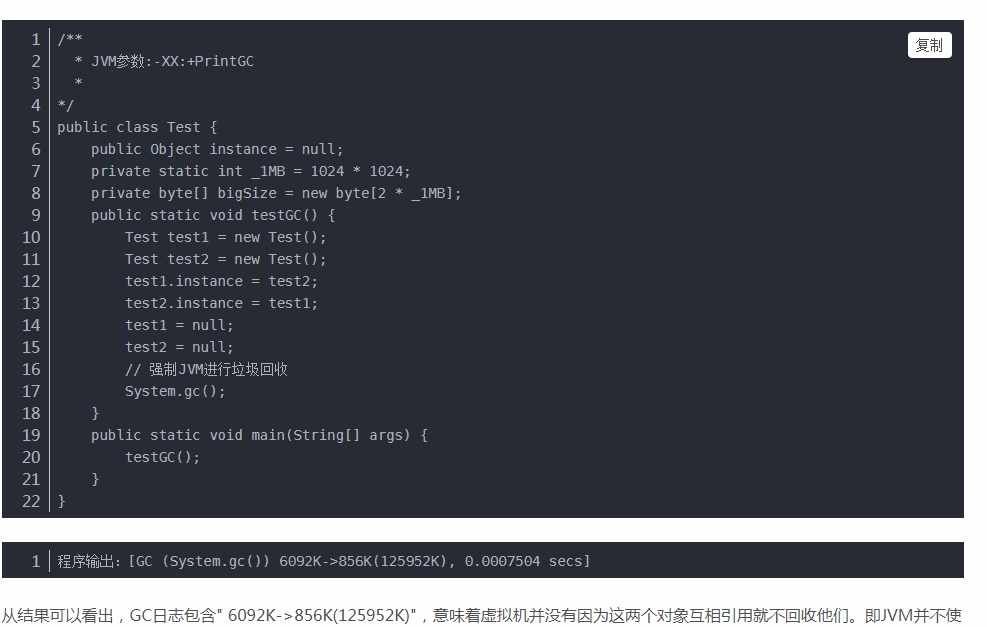
java堆中存放着几乎所有的对象实例，垃圾回收器在堆进行垃圾回收之前，首先要判断哪些对象还存活，哪些对象已经死去，可以回收。判断对象已死的算法有如下几种：

* 1. 引用计数算法：

描述：给对象增加一个引用计数器，每当有一个地方引用它时，计数器就+1；当引用失效时计数器-1；任何时刻的计数器为0的对象就是不能再被使用，即对象已死

引用计数法实现简单，判定的效率比较高，在大部分情况下都是一个比较好的算法。python语言就是采用的引用计数器算法来进行内存管理的。

但是在主流的JVM中没有选用引用计数法来管理内存，最主要的原因是引用计数法无法解决对象循环引用的问题。



* 1. 可达性分析算法

java采用可达性分析算法来判断对象是否存活

此算法的核心思想：通过一系列称为GC Roots 的对象作为起始点，从这些节点开始向下搜索，搜索走过的路径称为引用链，当一个对象到GC Roots 没有任何的引用链相连时（即从GC Roots到这个对象不可达），证明该对象不可用。

java中可作为GC Roots 的对象包含以下几种：

1. 虚拟机栈中引用的对象
2. 方法区中静态属性引用的对象
3. 方法区中常量引用的对象
4. 本地方法栈中引用的对象

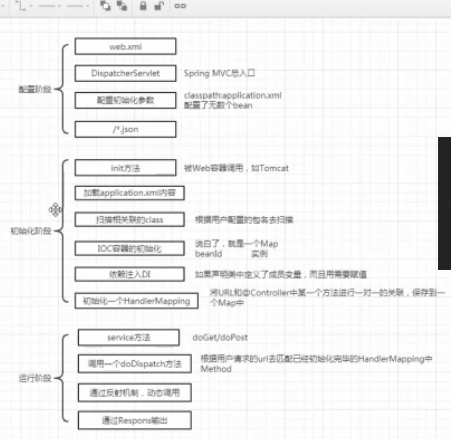
java中引用类型：强引用、软引用、弱引用、虚引用

三种基本的GC算法基石

1. GC标记\清除算法
2. GC复制算法
3. GC标记\整理算法

#### spring 原理

##### spring的三个阶段



配置阶段：

1、在web里面配置web.xml

2、DispatchServlet SpringMVC总入口

3、配置初始化参数 ： classpath：application.xml 配置Bean

4、配置一个URL： /\*.do

初始化阶段

1、init()方法 ：被web容器调用如tomcat等

2、加载application.xml

3、扫描相关联的class：根据用户配置的包名去扫描

4、IOC容器的初始化 ： 是一个Map beanID—实例

5、依赖注入（DI） ：如果声明的类中有成员变量，并且需要赋值，spring会自动赋值

6、初始化HandlerMapping： 将URL和Controller中的某一个方法进行一对一的对应，保存到一个Map中去

运行阶段：

1、service方法：doGet/doPost

2、调用一个doDispatch方法：根据用户请求的URL去匹配已经初始化完毕的HandlerMapping中的方法。

3、通过反射机制，动态调用方法，拿到返回值

4、通过response输出

## VUE学习

#### 简介

前端框架：

1. 单页面框架
2. 基于模块化组件化的开发模式

#### 搭建环境

1. 必须要安装nodejs

npm:node.js的一个包的安装工具命令，用的是外国的下载仓库

cnpm 淘宝镜像的包的安装工具命令，用的是国内淘宝的仓库

2、搭建vue的开发环境 ，安装vue的脚手架工具 官方命令行工具

npm install --global vue-cli / cnpm install --global vue-cli （此命令只需要执行一次）

3、创建项目 必须cd到对应的一个项目里面

vue init webpack vue-demo01

cd vue-demo01

cnpm install / npm install 如果创建项目的时候没有报错，这一步可以省略。如果报错了 cd到项目里面运行 cnpm install / npm install

npm run dev

## 设计模式学习

### 单例模式

#### 分类：懒汉式 饿汉式

#### 实现方法

1. 私有化构造方法
2. 创建一个静态方法来提供该单例对象

懒汉式：调用的时候再去创建对象，是线程不安全的

饿汉式：在类加载的时候去创建对象，是线程安全的

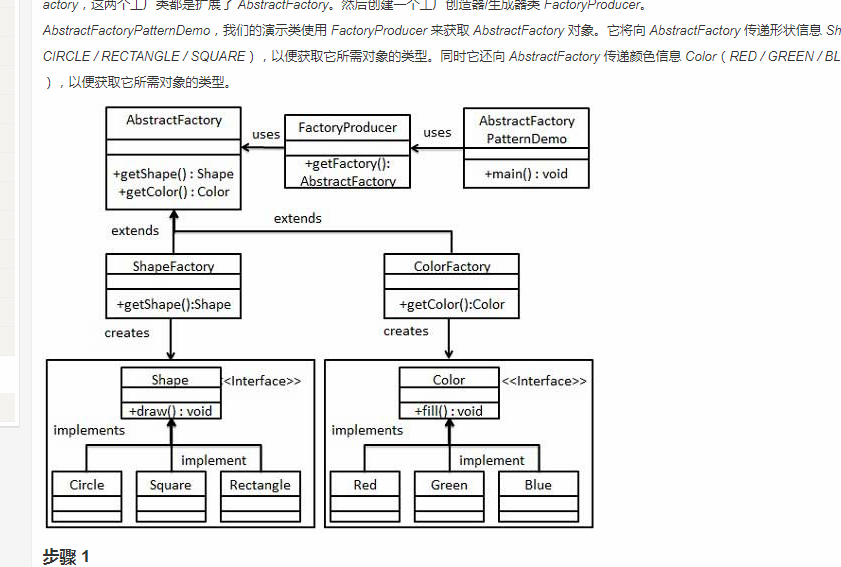
#### 应用场景：

许多时候整个系统只需要拥有一个的全局对象，这样有利于我们协调系统整体的行为

### 工厂方法模式：

工厂方法模式：工厂方法模式定义一个用于创建对象的接口，让子类决定实例化哪一个类。工厂方法使使一个类的实例化延迟到其子类。

#### 抽象工厂模式

抽象工厂就是在工厂模式的基础上在进行抽象

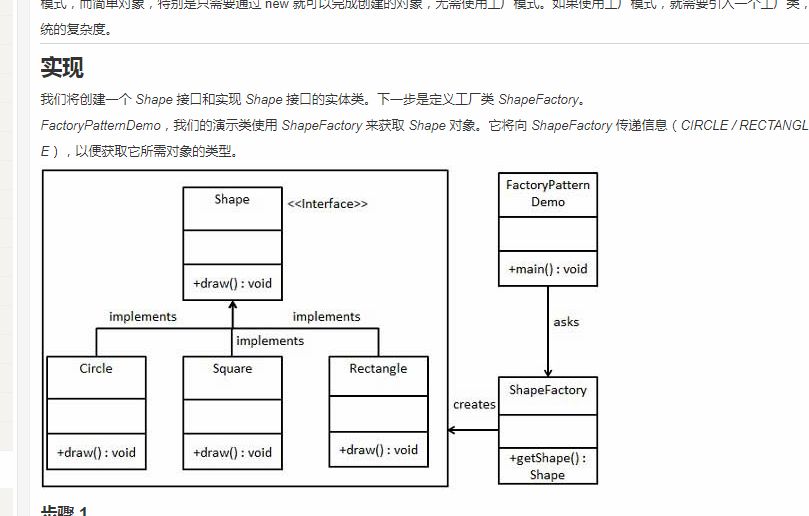
创建一个抽象类，定义获取工厂的方法

创建不同的工厂和方法

定义对象接口

定义实体类实现这个接口

#### 工厂模式



实现工厂模式的过程：

1. 抽象出来一个接口比如人类，动物类，定义抽象方法
2. 创建不同的对象实现这个接口
3. 创建一个工厂类，并创建一个工厂方法，该方法根据不同的需求创建不同的对象

4、在应用中需要创建对象的时候就可以创建工厂对象，调用方法，创建对象

### 生成器模式

### 策略模式

### 观察者模式