# 第一课

启动会，介绍公司情况及培训课程

# 第二课— Apache Commons

**基本内容：**Apache Commons包的用处，开源框架的好处，commons常用的工具类，以及介绍一些应用的场景①Map集合key和value的互取，通过BidiMap②相同key、value组合到collection,通过MulitMap;以及小点：编辑器找类和方法的快捷键，以及isEmpty()和isBlank()的区别，Map集合、Collection集合的特点等。

**学习体会：**基础很重要，每天稳固几个知识点，学习几个新知识点，坚持下去日积月累。下次上课提前预习些，要午睡。

个人认为在项目开发过程中最好能有这样的习惯：能用开源框架（开源框架基本都是众多程序员智慧的结晶，经得住考验）就尽量用，最大限度地减少编码量；即当编码处理一些业务逻辑时首先想想或找找有没相关的开源框架，有适合的就用。

重视Apache Commons

公司的各个应用下基本都有Apache Commons下的一些jar包，但在开发过程中习惯性用到这些工具包的童鞋可能比较少（经常使用到这些jar包的童鞋可以忽视），其实用好Apache Commons下工具集和帮助集可以减少好些编码量。 Apache Commons包含了很多开源的工具，用于解决平时编程经常会遇到的问题，减少重复编码。

## 2.1常用组件

BeanUtils 提供了对于JavaBean进行各种操作，克隆对象,属性等等.

对象克隆Person person2 = (Person)BeanUtils.cloneBean(person);将Map对象和Bean互相转化，BeanUtils.populate(person,map); Map map = BeanUtils.describe(person)

Betwixt XML与Java对象之间相互转换.

Collections java集合框架操作， 对java.util的扩展封装，处理数据还是挺灵活的。

org.apache.commons.collections – Commons Collections自定义的一组公用的接口和工具类

org.apache.commons.collections.bag – 实现Bag接口的一组类

org.apache.commons.collections.bidimap – 实现BidiMap系列接口的一组类

org.apache.commons.collections.buffer – 实现Buffer接口的一组类

org.apache.commons.collections.collection – 实现java.util.Collection接口的一组类

org.apache.commons.collections.comparators – 实现java.util.Comparator接口的一组类

org.apache.commons.collections.functors – Commons Collections自定义的一组功能类

org.apache.commons.collections.iterators – 实现java.util.Iterator接口的一组类

org.apache.commons.collections.keyvalue – 实现集合和键/值映射相关的一组类

org.apache.commons.collections.list – 实现java.util.List接口的一组类

org.apache.commons.collections.map – 实现Map系列接口的一组类

org.apache.commons.collections.set – 实现Set系列接口的一组类

Collection c = CollectionUtils.retainAll(list1, list2); //得到连个集合中相同的元素

BidiMap bidi = new TreeBidiMap(); BidiMap inverse = bidi.inverseBidiMap();

// returns a map with keys and values swapped

Configuration 一个java应用程序的配置管理类库.

DBCP 提供数据库连接池服务.

DbUtils 提供对jdbc 的操作封装来简化数据查询和记录读取操作.它是对JDBC的简单封装，对传统操作数据库的类进行二次封装，可以把结果集转化成List。，同时也不影响程序的性能。

FileUpload 提供文件上传功能.

Lang Java基本对象方法的工具类包 如：StringUtils,ArrayUtils等等.

。。。。。。

。。。。。

。。。

。。

。

## 2.2Map集合【拓展java集合系列--博客】

# 第三课-JAVA基础

## 3.1源码

1. Integer中有个静态内部类IntegerCache，里面有个cache[],也就是Integer常量池，常量池的大小为一个字节（-128~127）；当创建Integer对象时，不使用new Integer（int i）语句，大小在-128~127之间，对象存放在Integer常量池中。对于value在－128到127之间的int值，在调用这个方法创建Integer对象时，是直接从缓冲池中返回的，如果缓冲池中有，则返回的是同一个对象。
2. BigDecimal 我们的计算机是二进制的。浮点数没有办法是用二进制进行精确表示。我们的CPU表示浮点数由两个部分组成：指数和尾数，这样的表示方法一般都会失去一定的精确度，有些浮点数运算也会产生一定的误差。三种构造方法，int时使用参数为int的构造方法，其余采用string构造方法，) BigDecimal都是不可变的（immutable）的，在进行每一步运算时，都会产生一个新的对象，所以在做加减乘除运算时千万要保存操作后的值。

# 第四课IDE、GIT、MAVEN （商宇龙）

1. 程序员都要想着偷懒
2. 使用IDEA
3. GIT分支
4. MAVEN
5. GitHub的使用

**感悟：**差距真的很大，别人都用剩下的东西，你这还没学会，学习新的知识的能力，你的远见学识，拓宽你的吧，增加你获得知识的渠道，提高你的学习能力。

## Git

Git的优势在于跟踪并管理的是修改，而非文件。

Git信息配置（git config）

创建上传新项目（git init）

下载已有项目（git clone）

提交/修改文件（git commit）

查看日志（git log）

更新（git pull）

解决冲突

分支（git branch）

合并（git merge）

标签（git tag）

还原（git reset）

比对（git diff）

### 1.1命令：

查看配置情况：git config --list（cmd窗口或git GitBath窗口）

配置名字和邮箱：git config --global user.name “zhengzhongyi”

邮箱git config --global user.email [“zhenghzongyi@thunisoft.com”](mailto:\“zhenghzongyi@thunisoft.com\”)

Git创建本地版本库的命令是git init



2.1获取fetch/clone

获取是指本地版本库同步服务器更新的操作，此操作可使本地版本库得到最新的服务器信息，但无法改变当前的文件

2.2拉取pull

拉取是指本地工作区同步服务器更新，此操作可使本地工作区得到最新的服务器信息，并把有冲突的文件进行快速合并。

2.3注意：Git只能跟踪文本文件的改动，txt、网页、代码等，无法跟踪图片、视频这些二进制文件，包括word文件，所以要使用纯文本编写文件。Windows用户，不要使用**记事本**编辑任何文本文件，因为这个文件默认开头使用了十六进制的字符，编译会报错，建议使用notepad++替代记事本。

### 1.2实例

确定一个目录，右键Git Bash here

创建一个版本库

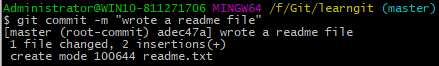


通过git init命令把这个目录变成Git可以管理的仓库



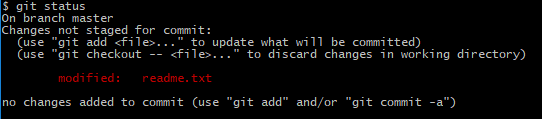
$ git add readme.txt 把文件添加到仓库

git commit -m "wrote a readme file" # -m后面是本次提交的说明



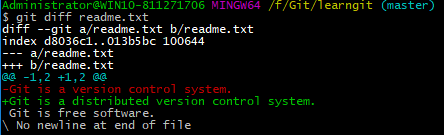
1 file changed：1个文件被改动（我们新添加的readme.txt文件）；2 insertions：插入了两行内容（readme.txt有两行内容）。

修改readme.txt文件，



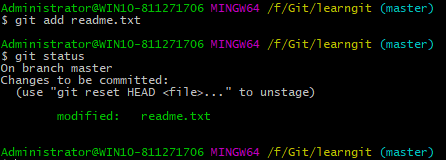
$git status，git status命令可以让我们时刻掌握仓库当前的状态，上面的命令输出告诉我们，readme.txt被修改过了，但还没有准备提交的修改。

$git diff

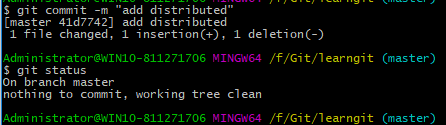


git diff顾名思义就是查看difference，显示的格式正是Unix通用的diff格式，可以从上面的命令输出看到，我们在第一行添加了一个distributed单词。

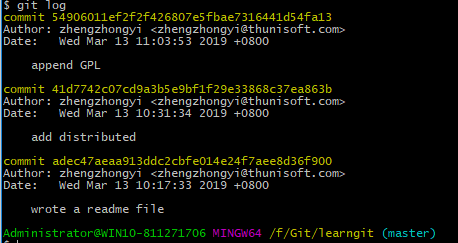
知道了对readme.txt做了什么修改以后，就可以放心提交到仓库了，提交修改和提交新文件是一样的两步，第一步 git add，没有任何输出，使用git statuse可以 查看当前仓库的状态，告诉我们提交的修改有哪些



第二步git commit,提交以后，git status显示当前仓库当前没有需要提交的修改，工作目录是干净的（working tree clean）

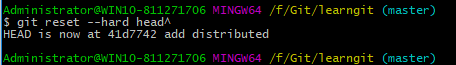


git log命令显示从最近到最远的提交日志

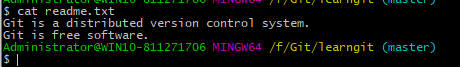


回到上个版本：Git必须知道当前版本是哪个版本，在Git中，用HEAD表示当前版本，也就是最新的提交，上一个版本就是HEAD^，上上一个版本就是HEAD^^。

$git reset --hard HEAD^

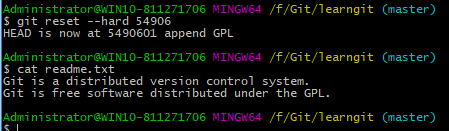


查看文件是否变了,$cat readme.txt



想再回到刚才那个版本：怎么办

只要上面的命令行窗口还没有被关掉，你就可以顺着往上找啊找啊，找到那个append GPL的commit id是1094adb...，于是就可以指定回到未来的某个版本：你看到的一大串类似1094adb...的是commit id（版本号），只需要写几位就可以



**现在总结一下：**

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。

穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本

### 1.3stage

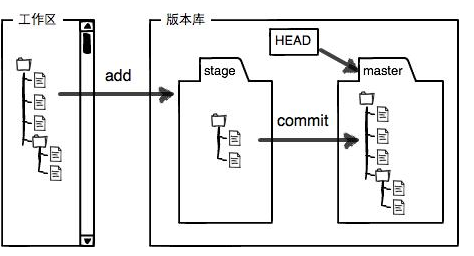
工作区（Working Directory）

就是你在电脑里能看到的目录，比如我的learngit文件夹就是一个工作区：

版本库（Repository）

工作区有一个隐藏目录.git，这个不算工作区，而是Git的版本库。

Git的版本库里存了很多东西，其中最重要的就是称为stage（或者叫index）的暂存区，还有Git为我们自动创建的第一个分支master，以及指向master的一个指针叫HEAD。



git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支。

使用git commit命令就是把暂存区的内容提交到分支，一旦提交，工作区就是干净的，暂存区是空白的

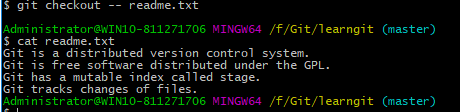
**撤销Git checkout --file**

命令git checkout -- readme.txt意思就是，把readme.txt文件在工作区的修改全部撤销，这里有两种情况：

一种是readme.txt自修改后还没有被放到暂存区，现在，撤销修改就回到和版本库一模一样的状态；

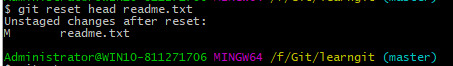
一种是readme.txt已经添加到暂存区后，又作了修改，现在，撤销修改就回到添加到暂存区后的状态。

总之，就是让这个文件回到最近一次git commit或git add时的状态。



撤销 Git reset HEAD file

git reset命令既可以回退版本，也可以把暂存区的修改回退到工作区。当我们用HEAD时，表示最新的版本

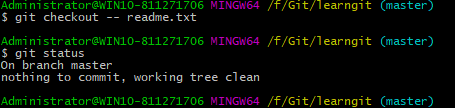


Git status

Changes to be committed:表示已经有内容通过工作区add到暂存区，目前暂存区有内容，待提交到分支

Changes not staged for commit:表示暂存区空白

**丢弃工作区的内容：$ git checkout -- readme.txt**



小结：

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD <file>，就回到了场景1，第二步按场景1操作。

删除：

$ rm test.txt删除工作区文件

$ git rm test.txt

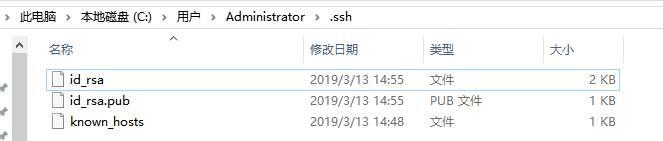
$ git commit -m "remove test.txt" 删除版本库中的文件 并提交

如果工作区删除了

$ git checkout -- test.txt 用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

### 1.4关联远程仓库

①$ ssh-keygen -t rsa -C "youremail@example.com"你需要把邮件地址换成你自己的邮件地址，然后一路回车，使用默认值即可，由于这个Key也不是用于军事目的，所以也无需设置密码。如果一切顺利的话，可以在用户主目录里找到.ssh目录，里面有id\_rsa和id\_rsa.pub两个文件，这两个就是SSH Key的秘钥对，id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人



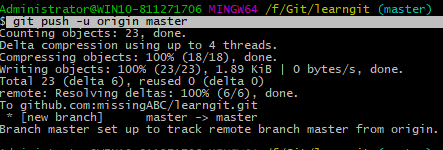
②第2步：登陆GitHub，打开“Account settings”，“SSH Keys”页面：

然后，点“Add SSH Key”，填上任意Title，在Key文本框里粘贴id\_rsa.pub文件的内容：

③在github上创建仓库，把本地仓库的内容推送到GitHub仓库$ git remote add origin git@github.com:github账户名/项目名.git

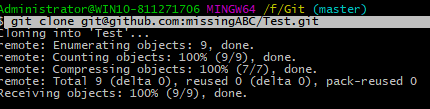
④以把本地库的所有内容推送到远程库上：

$ git push -u origin master



### 1.5从远程仓库克隆

$ git clone git@github.com:michaelliao/gitskills.git



### 1.6分支管理

创建dev分支

$ git checkout -b dev创建分支并切换到dev分支。相当于$ git branch dev和$ git checkout dev

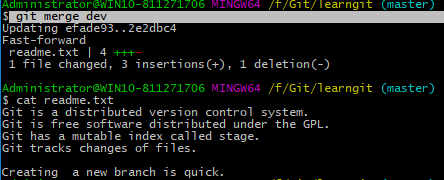
git branch命令会列出所有分支，当前分支前面会标一个\*号。

在dev分支上修改， $ git add readme.txt 和$ git commit -m "branch test"

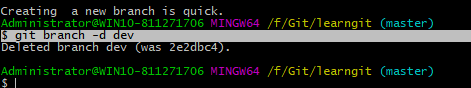
$git checkout master 切换回master主分支，cat readme.txt 发现文本内容没变

**合并分支到主分支**

$git merge dev



**删除dev分支**



小结：

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

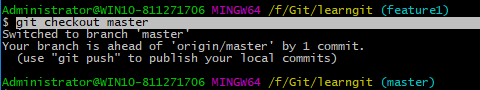
创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

**解决冲突：**

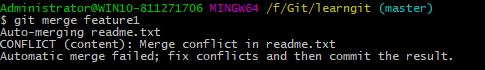
切换分支



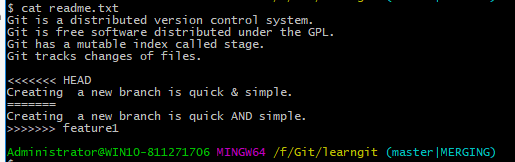
创建分支feature1 $ git checkout -b feature1

在feature1分支和master主分支上分别修改readme.txt，添加到暂存区并提交(不合并分支)

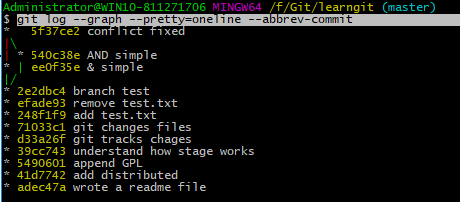
冲突产生了：这种情况下，Git无法执行“快速合并”，只能试图把各自的修改合并起来，但这种合并就可能会有冲突，



显示冲突内容



查看分支合并情况



删除分支

$ git branch -d feature1

Git分支十分强大，在团队开发中应该充分应用。

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

1.7Bug分支

Git还提供了一个stash功能，可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作：$ git stash，

$git stash list 一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

删除没有合并到master的分支,$ git branch -D feature-vulcan(分支名)

推送到远程库

$ git push origin master

### 1.7多人协作分支合并

分支提交到远程仓库出现冲突时，先git pull，然后再合并到远程，若git pull再出现冲突，则打开冲突的文件，内容会显示冲突的标识，进行手动修改，然后再合并到远程仓库。  
git pull命令的作用是：取回远程主机某个分支的更新，再与本地的指定分支合并。

一句话总结git pull和git fetch的区别：git pull = git fetch + git merge

1.8标签管理

进入分支，然后$ git tag v1.0 标签会自动记录时间 git tag 可以查看所有标签

标签默认是打在最新提交的commit id上的，若想打在昨天某时的，方法是找到历史提交的commit id 即可。$ git log --pretty=oneline --abbrev-commit 然后git tag v1.1 f52f35

# 星星点点

## Druid数据存储

-是一个分布式的支持实时分析的数据存储系统（Data Store）。Druid 设计之初的想法就是为分析而生，它在处理数据的规模、数据处理的实时性方面，比传统的OLAP 系统有了显著的性能改进，而且拥抱主流的开源生态，包括Hadoop 等。多年以来，Druid 一直是非常活跃的开源项目。

Druid 具有如下技术特点。

• 数据吞吐量大。

• 支持流式数据摄入和实时。

• 查询灵活且快。

• 社区支持力度大。

## 阿里Druid，

阿里巴巴开源平台上的一个项目，整个项目由数据库连接池、插件框架和SQL解析器组成。该项目主要是为了扩展JDBC的一些限制，可以让程序员实现一些特殊的需求，比如向密钥服务请求凭证、统计SQL信息、SQL性能收集、SQL注入检查、SQL翻译等，程序员可以通过定制来实现自己需要的功能。

## session为什么不能存大量数据？

### 3.1什么是session？

HTTP协议是无状态的，我们在浏览一个网站的时候 网站并不知道你是谁，为了解决这一点于是有了==会话管理==

会话管理主要包括cookie和session。cookie可以在客户端记录下信息，而session则是记录信息在服务器上。典型的应用场景：当我们登录一个网站，我们的登录信息就会存放在session当中，接下来在一些需要用户登录下才能操作的地方就可以通过session来判断用户的合法性。用户每次登录得到的==会话ID==是唯一的，这个会话ID可以保存在客户端中或者进行网络传输。客户端也就可以通过cookie或者url传输来传递这个会话ID获取存在服务器上的session信。

### 3.2session的原理

当程序需要为某个客户端的请求创建一个session的时候，服务器首先检查这个客户端的请求里是否已包含了一个session标识-称为sessionid，如果已包含一个sessionid则说明以前已经为此客户端创建过session，服务器就按照sessionid把这个session检索出来使用（如果检索不到，可能会新建一个），如果客户端请求不包含sessionid，则为此客户端创建一个session并且生成一个与此session相关联的sessionid，sessionid的值应该是一个既不会重复，又不容易被找到规律以仿造的字符串，这个sessionid将被在本次响应中返回给客户端保存。保存这个sessionid的方式可以采用cookie，这样在交互过程中浏览器可以自动的按照规则把这个标识发挥给服务器。

## SQL注入的防范

4.1简单有效的方法PreparedStatement

原理：sql注入只对sql语句的准备(编译)过程有破坏作用

而PreparedStatement已经准备好了,执行阶段只是把输入串作为数据处理,

而不再对sql语句进行解析,准备,因此也就避免了sql注入问题.

优点：

1).代码的可读性和可维护性.

(2).PreparedStatement尽最大可能提高性能.

(3).最重要的一点是极大地提高了安全性.

4.2使用正则表达式

4.3字符串过滤

4.4判断非法字符

## 日志级别

等级由低到高：debug<info<warn<Error<Fatal;

## 前端调试

Console.info()

# 六、测试