# 读书计划

**1. 深入理解Java虚拟机**

2. **HotSpot实战**

**3. Java并发编程实战**

**4. java多线程编程核心技术**

**5. java并发**

**6. Effective Java中文版**

**7. 深入分析Java Web技术内幕**

**8. 大型网站技术架构 核心原理与案例分析**

**9. 大型网站系统与Java中间件实践**

**10. 从Paxos到ZooKeeper 分布式一致性原理与实践**

**11. MySQL5.6从零开始学**

**11.13~12.13 Effective Java中文版**

3天1章

300 30 10

Mysql 账号root 密码 root

Windows service name: MySQL57

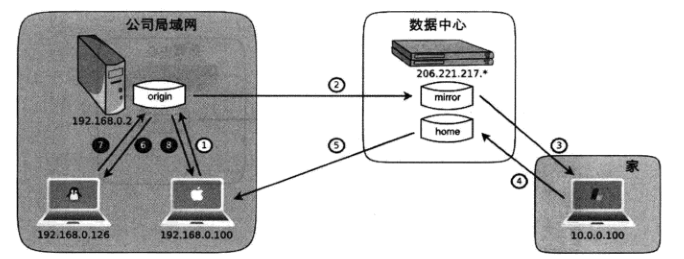
# Git

## Git初始化

### 摘要：git init ; git add; git commit; git log;

### 1.1为啥用Git?

1. 工作备份 公司本地推送到🡪公司git服务器🡪外网镜像自动备份
2. 异地协同 在家中推送到标记为home的版本库🡪从home库中同步到公司电脑

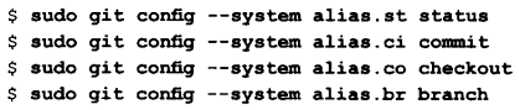


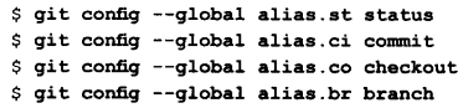
1. 现场版本控制
2. 重写提交说明
3. 误检入的文件的删除
4. 更好用的提交列表
5. 差异对比
6. 工作进度保存
7. 分页器
8. 快 **分布式控制系统**

### 1.2.Git初始化

创建版本库以及第一次提交

1. Git配置变量（全局文件或者系统文件）
   1. 用户，邮箱
   2. 别名

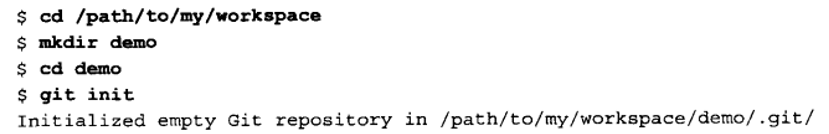




* 1. 开启颜色设置



1. 初始化版本库



路径/path/to/my/workspace为工作区根目录；

/path/to/my/workspace /demo为工作区；

隐藏.git目录就是Git的版本库，又叫仓库。

1. 在工作区添加文本文件，并添加到版本库
   1. $echo “Hello.” > welcome.txt //创建文件
   2. $git add welcome.txt //添加到仓库
   3. $git commit –m “initialized.” //ci是commit的别名,提交操作 –m 后为提交说明 必填

### 1.3.为什么工作区下会有一个.git目录？

分布式版本控制系统Git的一个**显著特点**，版本库位于工作区的根目录。

当在Git工作区的某个子目录下执行操作时，会在工作区目录中依次递归查找.git目录，否则就会报错

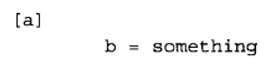
如何获取Git版本库的位置？



$ git rev-parse --git-dir

$ git rev-parse --show-toplevel

INI文件格式 节/键/值



git config命令的参数有何区别？



$ git config -e //版本库

$ git config -e --global //全局

$ git config -e --system //系统

读取配置

$ git config core.bare

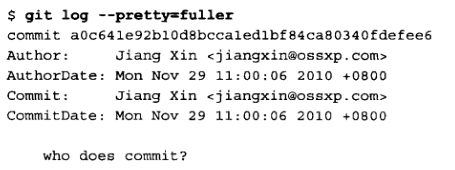
False

更改或设置

$ git config a.b.c = something

### 1.4.是谁完成的提交？

#### 1.4.1.通过日志查看作者和提交者



#### 1.4.2.修改作者和提交者信息

$ git commit –amend –allow –empty – reset-author

--amend 对刚刚提交的进行修补

--allow-empty 空白提交被允许

--reset-author 同步修改autor id 不然只会修改commit的id

## Git暂存区

摘要：git status; git diff; git stash;

### 修改不能直接提交么？

必须先add后commit

#### 2.1.1查看状态



M位于第一列：说明**版本库**中的文件与**暂存区**的中的文件不同；

M位于第二列：说明**本地**工作区中的文件与**暂存区**的文件不同；

#### 2.1.2对比差异

$ git diff 工作区与暂存区相对比

$ git diff HEAD 工作区与当前工作分支对比

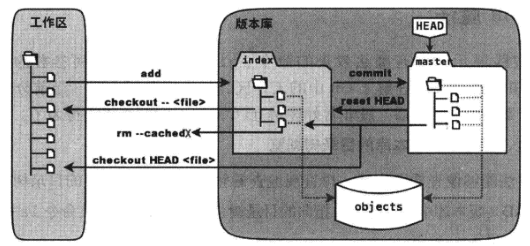
$ git diff --cached 暂存区与版本库的对比

#### 2.1.3保存工作进度

$ git status

### 理解Git暂存区

扫描工作区更改时先依据.git/index中的时间戳对比，时间戳变了文件内容可能改变



#### 理解工作区与暂存区版本库之间的数据流：

Head是指向maste的一个游标

**工作区🡪暂存区**

git add,暂存区的目录将被更新，工作区修改的内容会被写到一个新的对象库，对象ID被写入暂存区文件索引

**暂存区🡪版本库**

git commit

**版本库🡪暂存区**

git checkout head 将版本库的文件替换暂存区文件

git reset Head 暂存区的目录树会被重写

**暂存区🡪工作区**

git checkout

**从暂存区移除**

Git Rm --cahched

#### 浏览工作区，暂存区，版本库目录树

查看工作区的文件

$ find . -path ./.git -prune -o -type f -printf "%-20p\t%s\n"

查看暂存区的文件

$ git ls-files –s

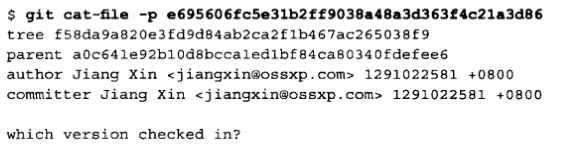
如果要针对暂存区使用git ls-tree 需要先执行 git write-tree 先将目录树写入

查看版本库的文件

$ git ls-tree -l head

#### Head和Master到底是什么东西？

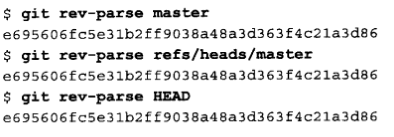
 🡪 🡪

🡪 

Master指向的是一个commit ID 这样就可以获得历史跟踪链

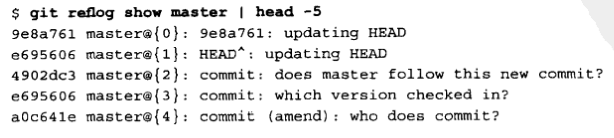
目录.git/ref是保存引用的命名空间

.git/ref/heads目录下的引用又称为分支



## Git重置

### 3.1.挽救错误的重置



### 3.2.深入git reset



#### 3.2.1.用法一（替换暂存区）

引用回退，暂存区回退（等价 –mixed）

Git reset（等价Git reset head） 仅用head指向的目录树重置暂存区，工作区不会受影响 相当于撤销之前的git add 引用不改变

Git reset head^ 用上次的提交替换缓存区

**反向操作**（git add filename）

Git reset **–filename**

Git reset head filename

#### 3.2.2用法二(soft,mixed,hard)

只更改引用的指向（**head**为指向master的引用；**head^**表示上一次的提交）

Git reset –soft <commit> (commit 可以为**hash值** 也可以为**master@{}**)

更改引用指向，重置暂存区

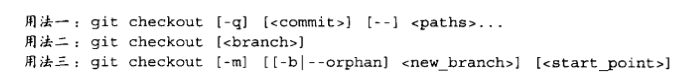
Git reset –mixed head^

更改引用指向，重置暂存区，重置工作区

Git reset –hard head^

## Git检出

深入了解checkout



用法一：**重置**的默认值是**head** ，**检出**的默认值是**暂存区**

重置一般重置暂存区， 检出一般覆盖工作区

用法二：切换分支 如果省略branch 相当于对工作区进行状态检查

Git checkout branch

检出branch分支，用branch指向的树更新暂存区和工作区

Git checkout

Git checkout head

## Git删除

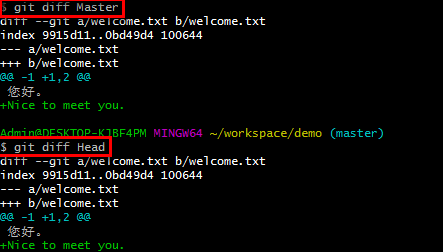
删除本地

Rm

删除本地和暂存区

Git rm

### 疑问



恢复进度

mvn archetype:generate -DgroupId=com.yiibai -DartifactId=NumberGenerat or -DarchetypeArtifactId=maven -archetype-quickstart -DinteractiveMode=false

# Idea

## 1.快捷键

|  |
| --- |
| [IntelliJ Idea 常用快捷键列表](http://www.cnblogs.com/zhangpengshou/p/5366413.html) Ctrl+Shift + Enter，语句完成 “！”，否定完成，输入表达式时按 “！”键 Ctrl+E，最近的文件 Ctrl+Shift+E，最近更改的文件 Shift+Click，可以关闭文件 Ctrl+[ OR ]，可以跑到大括号的开头与结尾 Ctrl+F12，可以显示当前文件的结构 Ctrl+F7，可以查询当前元素在当前文件中的引用，然后按 F3 可以选择 Ctrl+N，可以快速打开类 Ctrl+Shift+N，可以快速打开文件 Alt+Q，可以看到当前方法的声明 Ctrl+P，可以显示参数信息 Ctrl+Shift+Insert，可以选择剪贴板内容并插入 Alt+Insert，可以生成构造器/Getter/Setter等 Ctrl+Alt+V，可以引入变量。例如：new String(); 自动导入变量定义 Ctrl+Alt+T，可以把代码包在一个块内，例如：try/catch Ctrl+Enter，导入包，自动修正 Ctrl+Alt+L，格式化代码 Ctrl+Alt+I，将选中的代码进行自动缩进编排，这个功能在编辑 JSP 文件时也可以工作 Ctrl+Alt+O，优化导入的类和包 Ctrl+R，替换文本 Ctrl+F，查找文本 Ctrl+Shift+Space，自动补全代码 Ctrl+空格，代码提示（与系统输入法快捷键冲突） Ctrl+Shift+Alt+N，查找类中的方法或变量 Alt+Shift+C，最近的更改 Alt+Shift+Up/Down，上/下移一行 Shift+F6，重构 – 重命名 Ctrl+X，删除行 Ctrl+D，复制行 Ctrl+/或Ctrl+Shift+/，注释（//或者/\*\*/） Ctrl+J，自动代码（例如：serr） Ctrl+Alt+J，用动态模板环绕 Ctrl+H，显示类结构图（类的继承层次） Ctrl+Q，显示注释文档 Alt+F1，查找代码所在位置 Alt+1，快速打开或隐藏工程面板 Ctrl+Alt+left/right，返回至上次浏览的位置 Alt+left/right，切换代码视图 Alt+Up/Down，在方法间快速移动定位 Ctrl+Shift+Up/Down，向上/下移动语句 F2 或 Shift+F2，高亮错误或警告快速定位 Tab，代码标签输入完成后，按 Tab，生成代码 Ctrl+Shift+F7，高亮显示所有该文本，按 Esc 高亮消失 Alt+F3，逐个往下查找相同文本，并高亮显示 Ctrl+Up/Down，光标中转到第一行或最后一行下 Ctrl+B/Ctrl+Click，快速打开光标处的类或方法（跳转到定义处） Ctrl+Alt+B，跳转到方法实现处 Ctrl+Shift+Backspace，跳转到上次编辑的地方 Ctrl+O，重写方法 Ctrl+Alt+Space，类名自动完成 Ctrl+Alt+Up/Down，快速跳转搜索结果 Ctrl+Shift+J，整合两行 Alt+F8，计算变量值 Ctrl+Shift+V，可以将最近使用的剪贴板内容选择插入到文本 Ctrl+Alt+Shift+V，简单粘贴 Shift+Esc，不仅可以把焦点移到编辑器上，而且还可以隐藏当前（或最后活动的）工具窗口 F12，把焦点从编辑器移到最近使用的工具窗口 Shift+F1，要打开编辑器光标字符处使用的类或者方法 Java 文档的浏览器 Ctrl+W，可以选择单词继而语句继而行继而函数 Ctrl+Shift+W，取消选择光标所在词 Alt+F7，查找整个工程中使用地某一个类、方法或者变量的位置 Ctrl+I，实现方法 Ctrl+Shift+U，大小写转化 Ctrl+Y，删除当前行  Shift+Enter，向下插入新行 psvm/sout，main/System.out.println(); Ctrl+J，查看更多 Ctrl+Shift+F，全局查找 Ctrl+F，查找/Shift+F3，向上查找/F3，向下查找 Ctrl+Shift+S，高级搜索 Ctrl+U，转到父类 Ctrl+Alt+S，打开设置对话框 Alt+Shift+Inert，开启/关闭列选择模式 Ctrl+Alt+Shift+S，打开当前项目/模块属性 Ctrl+G，定位行 Alt+Home，跳转到导航栏 Ctrl+Enter，上插一行 Ctrl+Backspace，按单词删除 Ctrl+”+/-”，当前方法展开、折叠 Ctrl+Shift+”+/-”，全部展开、折叠 【调试部分、编译】 Ctrl+F2，停止 Alt+Shift+F9，选择 Debug Alt+Shift+F10，选择 Run Ctrl+Shift+F9，编译 Ctrl+Shift+F10，运行 Ctrl+Shift+F8，查看断点 F8，步过 F7，步入 Shift+F7，智能步入 Shift+F8，步出 Alt+Shift+F8，强制步过 Alt+Shift+F7，强制步入 Alt+F9，运行至光标处 Ctrl+Alt+F9，强制运行至光标处 F9，恢复程序 Alt+F10，定位到断点 Ctrl+F8，切换行断点 Ctrl+F9，生成项目 Alt+1，项目 Alt+2，收藏 Alt+6，TODO Alt+7，结构 Ctrl+Shift+C，复制路径 Ctrl+Alt+Shift+C，复制引用，必须选择类名 Ctrl+Alt+Y，同步 Ctrl+~，快速切换方案（界面外观、代码风格、快捷键映射等菜单） Shift+F12，还原默认布局 Ctrl+Shift+F12，隐藏/恢复所有窗口 Ctrl+F4，关闭 Ctrl+Shift+F4，关闭活动选项卡 Ctrl+Tab，转到下一个拆分器 Ctrl+Shift+Tab，转到上一个拆分器 【重构】 Ctrl+Alt+Shift+T，弹出重构菜单 Shift+F6，重命名 F6，移动 F5，复制 Alt+Delete，安全删除 Ctrl+Alt+N，内联 【查找】 Ctrl+F，查找 Ctrl+R，替换 F3，查找下一个 Shift+F3，查找上一个 Ctrl+Shift+F，在路径中查找 Ctrl+Shift+R，在路径中替换 Ctrl+Shift+S，搜索结构 Ctrl+Shift+M，替换结构 Alt+F7，查找用法 Ctrl+Alt+F7，显示用法 Ctrl+F7，在文件中查找用法 Ctrl+Shift+F7，在文件中高亮显示用法 |

# Web

## Cookie与Session

参考：

<http://blog.csdn.net/fangaoxin/article/details/6952954/>

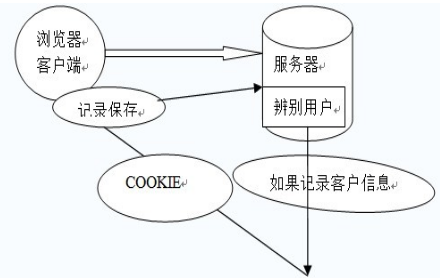
俩种常用的会话跟踪技术

cookie通过在客户端记录信息确定用户身份；

session通过在服务端记录信息确定用户身份；

Http协议是一种无状态的协议。一旦数据交换完成，服务端与客户端的连接就会关闭，再次数据交换重新建立连接。意味着无法跟踪上次的会话。

Cookie正是为了弥补此不足。



Cookie就像是服务端颁发给客户端的通行证，正是通过此通行证，服务端可以辨别用户的身份。

一个域名对应着一个cookie,不能跨域名访问cookie。

### Cookie的重要属性：

#### Int maxAge

正数：多少秒之后失效

0：删除

负数：临时cookie

#### 域名：

同一个一级域名下的两个二级域名如www.helloweenvsfei.com和images.helloweenvsfei.com如果想所有

helloweenvsfei.com名下的二级域名都可以使用该Cookie，需要设置Cookie的domain参数

cookie.setDomain(".helloweenvsfei.com");           // 设置域名

#### 路径：

domain属性决定运行访问Cookie的域名，而path属性决定允许访问Cookie的路径（ContextPath）。例如，如果只允许/sessionWeb/下的程序使用Cookie，可以这么写：

Cookie cookie = new Cookie("time","20080808");     // 新建Cookie

cookie.setPath("/session/");                          // 设置路径

response.addCookie(cookie);                           // 输出到客户端

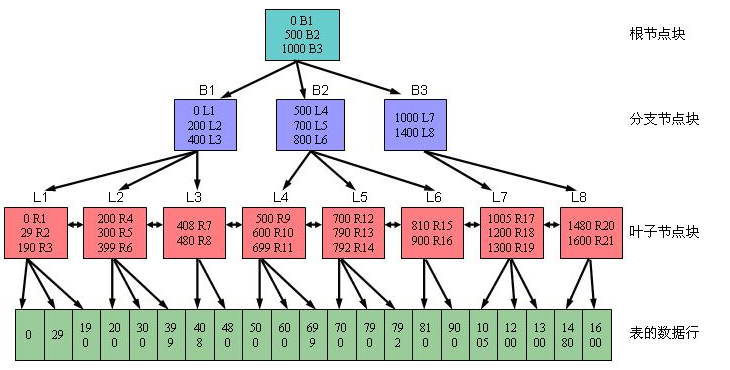
设置为“/”时允许所有路径使用Cookie。path属性需要使用符号“/”结尾。name相同但domain相同的两个Cookie也是两个不同的Cookie。

# Sql

## 1. 索引

### 1.1 B-Tree

#### 图解



#### 原理

索引条目都是顺序排列。

分支节点----每个索引包含俩个字段：1.最小键，2.下一索引地址

叶节点 ----俩个字段：1.最小键2.RowID 记录行的物理地址

#### 创建

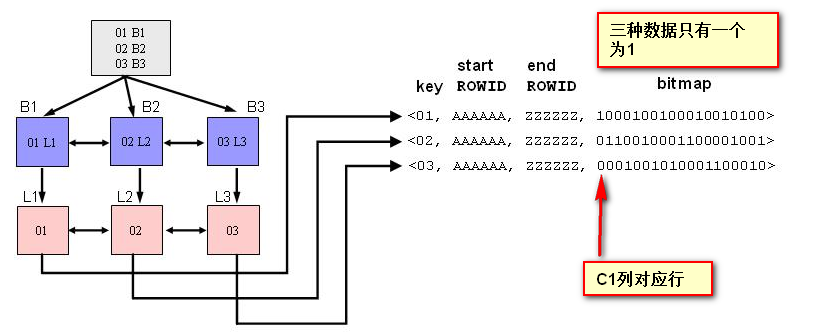
　 --创建索引  
　　create index idx\_emp1\_ename on emp1(ename);  
　　--创建唯一索引  
　　create unique index idx\_uq\_emp1\_empno on emp1(empno) tablespace mypl;

#### 参考

<https://www.cnblogs.com/iliuyuet/p/4431464.html>

### BitMap

#### 图解



#### 原理

Bitmap 1010001001001 根据start rowid+行数获取当前数据rowid

备注：1，2，3在该行存在置为1否则0

Bitmap索引适合**重复值**很多的列，不适合经常**update**的列

#### 创建

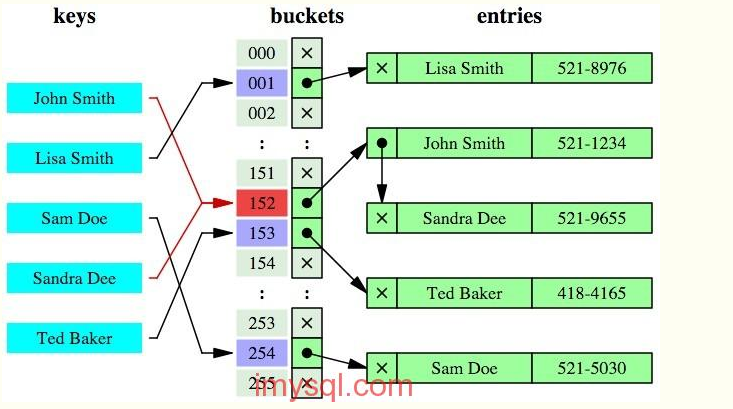
create bitmap index idx\_bm\_emp1\_deptno on emp1(deptno);

#### 参考

<http://www.cnblogs.com/LBSer/p/3322630.html>

### 1.3 Hash

#### 图解



#### 原理

* SQL Server 拥有一个用于所有哈希索引的哈希函数。
* 哈希函数具有确定性。 同一索引键始终映射到哈希索引中的同一 Bucket。
* 多个索引键可能映射到同一个哈希 Bucket。
* 哈希函数经过均衡处理，这意味着索引键值在哈希桶上的分布通常符合泊松分布。

泊松分布并非均匀分布。 索引键值并非均匀地分布在哈希 Bucket中。 例如，泊松分布的 *n* 非重复索引键通过 *n* 哈希桶中约三分之一空存储桶，包含一个索引键，则存储桶的三分之一的结果和其他第三个包含两个索引键。 少量 Bucket 将包含两个以上的键。

#### 创建

例：UPDATE 表名 SET 表名.url\_src = SRC32（表名.url）;  
表中数据是这样的：

|  |  |
| --- | --- |
| URL | url\_src |
| http://www.mysql.com | 1560514994 |

这样，查询就可以进行优化了:  
SELECT ID FROM 表名WHERE URL="http://www.mysql.com" AND url\_src=SRC32("http://www.mysql.com");

1.4 聚族索引

1.5 非聚族索引

## 2. plSql配置

### 2.1配置环境变量

1) 变量名：ORACLE\_HOME  变量值：C:\oraclexe\app\oracle\product\11.2.0\server\

2) 变量名：TNS\_ADMIN  变量值：C:\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN

3) 变量名：NLS\_LANG   变量值：SIMPLIFIED CHINESE\_CHINA.ZHS16GBK

4) 修改Path变量，在后面添加 C:\oraclexe\app\oracle\product\11.2.0\server\bin

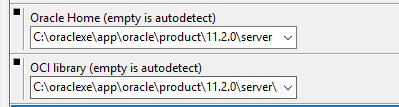
### 2.2listener.ora

|  |
| --- |
| LISTENER =  (DESCRIPTION\_LIST =  (DESCRIPTION =  (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1))  (ADDRESS = (PROTOCOL = TCP)(HOST = JM)(PORT = 1521))  )  ) |

### 2.3tnsnames.ora

|  |
| --- |
| XE =  (DESCRIPTION =  (ADDRESS = (PROTOCOL = TCP)(HOST = JM)(PORT = 1521))  (CONNECT\_DATA =  (SERVER = DEDICATED)  (SERVICE\_NAME = XE)  )  ) |

### 2.4Tools🡪preferences



## 3. oracle 创建用户

|  |
| --- |
| *--创建临时表空间*  create temporary tablespace jm\_temp  tempfile 'C:/oraclexe/app/oracle/oradata/JM/jm\_temp.dbf'  size 32m  autoextend on  next 32m maxsize 2048m  extent management local;  *--创建数据表空间*  create tablespace jm\_data  datafile 'C:/oraclexe/app/oracle/oradata/JM/jm\_data.dbf'  size 32m  autoextend on  next 32m maxsize 2048m  extent management local;  *--创建用户并指定表空间*  create user jm identified by jm  default tablespace jm\_data  temporary tablespace jm\_temp;  *--给用户授予权限*  grant connect,resource to jm;  *--以后以该用户登录，创建的任何数据库对象都属于test\_temp 和test\_data表空间，这就不用在每创建一个对象给其指定表空间了。*  *-- 分配用户 jm 创建表，创建序列，创建存储过程和创建视图的权限*  grant create table,create sequence,create view,create procedure to jm; |

# Java

## JVM

参考

http://blog.csdn.net/qq\_34107571/article/details/78151314?locationNum=2&fps=1

## 乱码

### 2.1 response乱码

1. 设置服务器端的编码

response.setCharacterEncoding("utf-8”);

1. 通知浏览器服务器发送的数据格式

response.setContentType("text/html;charset=utf-8”);

1. 浏览器使用utf-8进行解码

### request乱码

1. post方式

request.setCharacterEncoding("utf-8”)

1. get方式

get方式提交的参数会跟在请求行中的uri后边，服务器按照默认的iso-8859-1进行解码，这时候解决乱码有两种办法：

办法一：修改服务器端对uri参数的默认编码

在tomcat的server.xml中，设置元素的属性URIEncoding="UTF-8”即可。（默认没有设置此属性）

**(上述方法不实际)**

方法二：String str = new String(request.getParameter("参数").getBytes("ISO-8859-1"), "UTF-8");

# Effective Java

1. 创建和销毁对象

1.1 考虑用静态工厂方法替代构造器

优势：

1. 有名称，语义更清晰（参数顺序不同的俩个构造器，该用哪个？？？）
2. 不必每次调用都创建对象（如此可以使用a==b来判断对象相同，提高性能，废话！本来就是一个对象（枚举类型））
3. 可以返回原类型的任何子类型

通过接口引用返回对象

接口

注册

服务访问api

服务提供者接口

可不可以这样理解，针对接口编程，我们不清楚具体的实现类是哪个?这个时候就要指定一个实现类—这里通过不可实例化的注册类，此类中含有静态工厂方法，指定返回类型，即具体的实现类。

2. 对所有对象都通用的方法

# Linux

## Centos网络配置

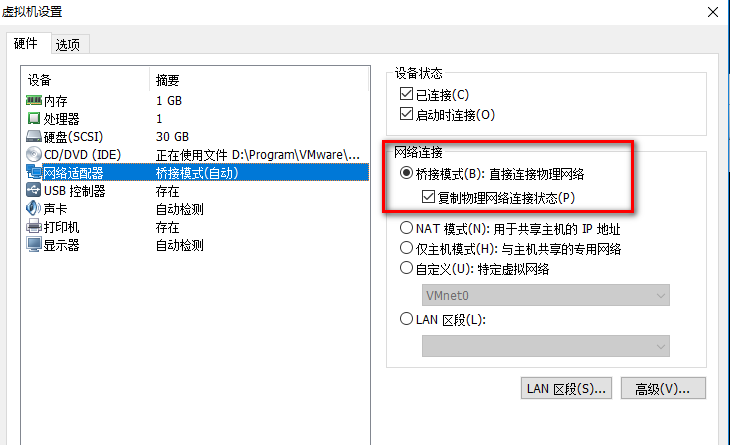
### 1.1.编辑ifcfg-eth0

路径：/etc/sysconfig/network-scripts/ifcfg-eth0

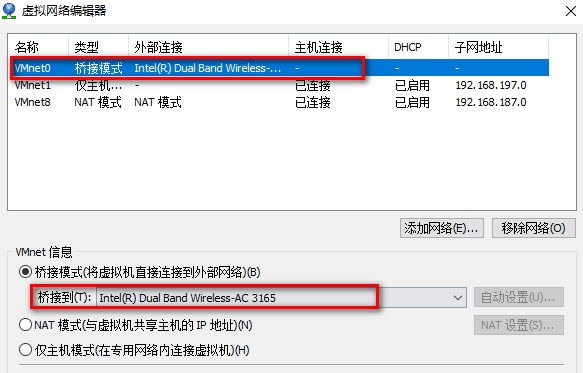
|  |
| --- |
| DEVICE=eth0  TYPE=Ethernet  UUID=00cdfec4-957f-4ca9-afe7-a7a81a8d8a10  ONBOOT=yes  NM\_CONTROLLED=yes  BOOTPROTO=none  HWADDR=00:0C:29:76:31:0D  IPADDR=10.136.192.20  PREFIX=22  GATEWAY=10.136.195.254  DNS1=223.5.5.5  DEFROUTE=yes  IPV4\_FAILURE\_FATAL=yes  IPV6INIT=no  NAME="System eth0" |

### 1.2.网络连接方式：为桥接模式

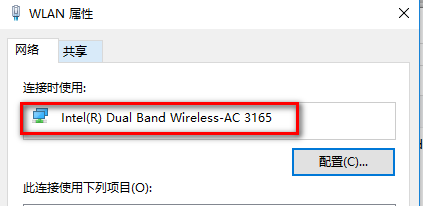
虚拟机🡪设置



### 1.3.桥接到哪？



备注：可以在当前网络的属性里查看



### 1.4.关闭防火墙

关闭：service iptables stop

永久关闭：chkconfig iptables off（要执行reboot）

查看状态 service iptables status

备注：

重启网卡

service network restart

查看网络配置

Ifconfig

# Spring boot

## 1.日志

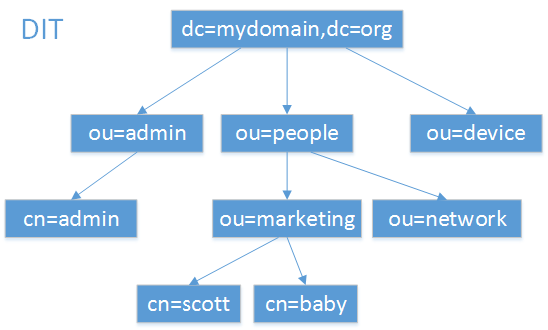
<http://tengj.top/2017/04/05/springboot7/>

## Ldap

[LDAP](http://www.openldap.org/doc/admin24/intro.html)是轻量目录访问协议(Lightweight Directory Access Protocol)的缩写。

**特点：**

* LDAP的结构用树来表示，而不是用表格。正因为这样，就不能用SQL语句了
* LDAP可以很快地得到查询结果，不过在写方面，就慢得多
* Client/server模型，Server 用于存储数据，Client提供操作目录信息树的工具
* **这些工具可以将数据库的内容以文本格式（LDAP 数据交换格式，LDIF）呈现在您的面前**



**基本概念：**

* Entry(条目)

数据库中的记录，每一个条目都有唯一的标识名dn

dn："cn=baby,ou=marketing,ou=people,dc=mydomain,dc=org"

* Attribute(属性)

每个条目可有很多属性

* ObjectClass(对象)

属性集合

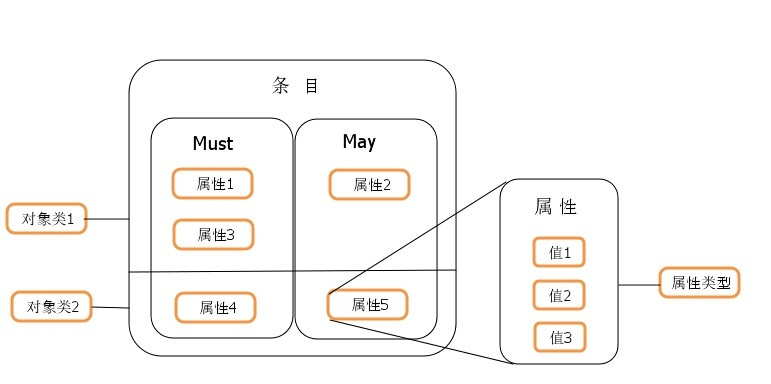
* Schema(模式)

对象类（ObjectClass）、属性类型（AttributeType）、语法（Syntax）分别约定了条目、属性、值，他们之间的关系如下图所示。所以这些构成了模式(Schema)——对象类的集合。

**注意：**

条目数据在导入时通常需要接受模式检查，它确保了目录中所有的条目数据结构都是一致的。

Must是必要属性，may可有可无



dn: uid=ben,ou=people,dc=springframework,dc=org  
objectclass: top  
objectclass: person  
objectclass: organizationalPerson  
objectclass: inetOrgPerson  
cn: Ben Alex  
sn: Alex  
uid: ben  
userPassword: {SHA}nFCebWjxfaLbHHG1Qk5UU4trbvQ=

属性：

cn commonName 姓名

sn surname 姓

uid userid 用户名

userPassword 密码

**总结：**

entry（唯一的dn）有很多属性，继承自objectclass，objectClass定义在schema中。entry条目进来后会接受schema模式的检查。

OpenLdap 下载地址

Windows:

<https://www.userbooster.de/download/openldap-for-windows.aspx>

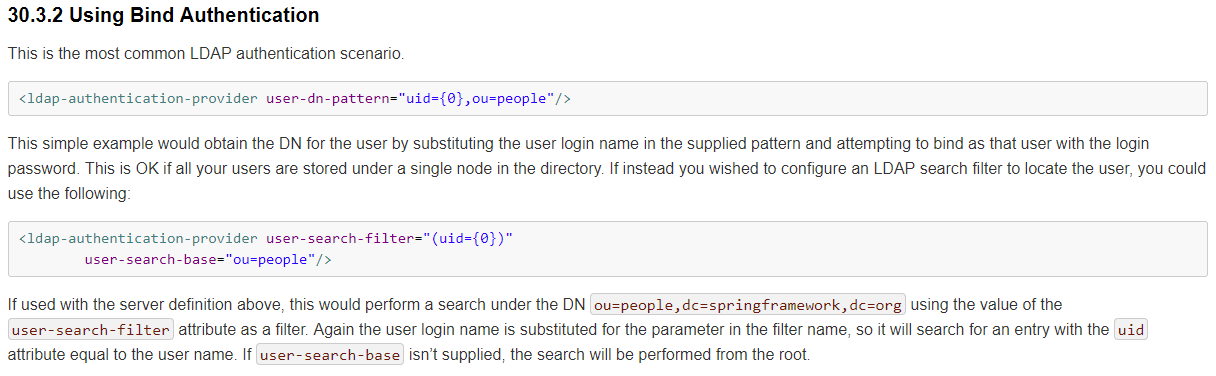
linux:

slapd.exe -d 1 -f ./slapd.conf

slapadd -l ./ldifdata/test.ldif.txt

#spring.ldap.embedded.ldif=classpath:test-server.ldif  
#spring.ldap.embedded.base-dn=dc=maxcrc,dc=com  
#spring.ldap.embedded.port=389

ldap.contextSource.url=ldap://10.136.192.208:389  
ldap.contextSource.base=dc=maxcrc,dc=com  
ldap.contextSource.userDn=cn=Manager,dc=maxcrc,dc=com  
ldap.contextSource.password=roots



|  |
| --- |
| Either an LdapUserSearch or DN pattern (or both) must be supplied |

Principle(User), Authority(Role)和Permission是Spring Security的3个核心概念。

权限加载：

通过以下属性控制：

--group-search-base

--group-role-attribute

--group-search-filter

俩种验证策略

1. Authentication directly to the LDAP server

2. Password comparison

## Spring security

LDAPAuthnConfiguration

自定义 Filter 以及相关辅助类，实现用户、角色、权限、资源的数据库管理，涉及相关接口或类说明如下：

1）   AbstractSecurityInterceptor

具体实现类作为过滤器，该过滤器要插入到授权之前。在执行 doFilter 之前，进行权限的检查，而具体的实现交给 accessDecisionManager 。

2）   FilterInvocationSecurityMetadataSource

具体实现类在初始化时，要实现从数据库或其它存储库中加载所有资源与权限（角色），并装配到 MAP <String, Collection<ConfigAttribute>> 中。 资源通常为 url ， 权限就是那些以 ROLE\_ 为前缀的角色，资源为 key ， 权限为 value 。 一个资源可以由多个权限来访问。

3）   UserDetailService

具体实现类从存储库中读取特定用户的各种信息（用户的密码，角色信息，是否锁定，账号是否过期等）。唯一要实现的方法： public UserDetails loadUserByUsername(String username)

4）   AccessDecisionManager

匹配权限以决定是否放行。主要实现方法： **public void**decide (Authentication authentication, Object object,

           Collection<ConfigAttribute> configAttributes)

//In this method, need to compare authentication with configAttributes.

Ø   A object is a URL, a filter was find permission configuration by this URL, and pass to here.

Ø     Check authentication has attribute in permission configuration (configAttributes)

Ø   If not match corresponding authentication, throw a AccessDeniedException.

**2.6 Spring Security 评价**

在 Spring Security 世界里，可以区分出哪些资源可以匿名访问，哪些需要角色权限，哪个页面提供登录功能；怎样进行用户身份认证，用户的密码如何加密。哪些资源必须使用 https 协议，资源和访问端口有怎样的对应关系。

下面就优点和缺点对 Spring Security 进行点评。

1、 **优点**

总体说来 Spring Security 具有以下几个优点：

1） 提供了一套权限框架，这套框架是可行的；

**2）**提供了很多用户身份认证功能，可以节约大量开发工作；

**3）**提供了角色判断功能，这点既是优点又是缺点；

**4）**提供了 form-login 、 remember me 等控制。

其中 2 、 4 两点，对于我们中国开发者，可用性并不大。我们的系统大多采用用户名 / 密码身份认证模式，大多公司都有可复用代码。 form-login 、 remember me 等这些功能，并不是难以开发，而且可用之处也并不多。

## [spring boot 之热部署(三)](http://www.cnblogs.com/bingshu/p/6876030.html)

       热部署：当发现程序修改时自动启动应用程序。

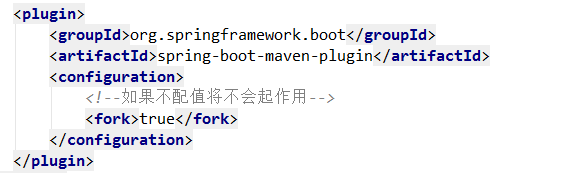
spring boot使用的是spring-boot-devtools是一个为开发者服务的一个模块。其原理用了classLoader 其中一个加载不变的类，另一个restart ClassLoader加载变得类。

devtools会监听classpath下的文件变动，并且会立即重启应用（发生在保存时机）。

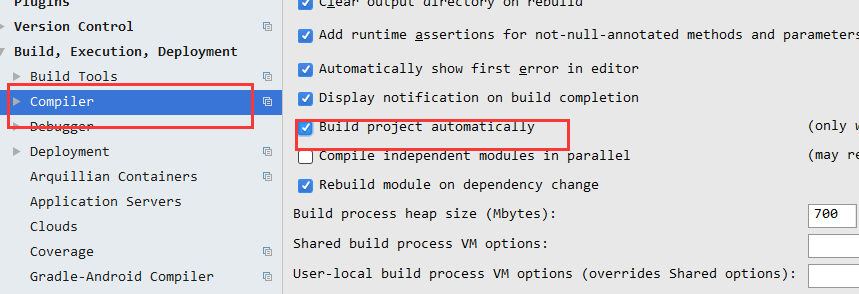
  一、pom的依赖直接添加坐标。



二、编译节点添加。



三、（1）设置Idea   Ctrl+Alt+S



（2）Ctrl+Shift+Alt+ /

