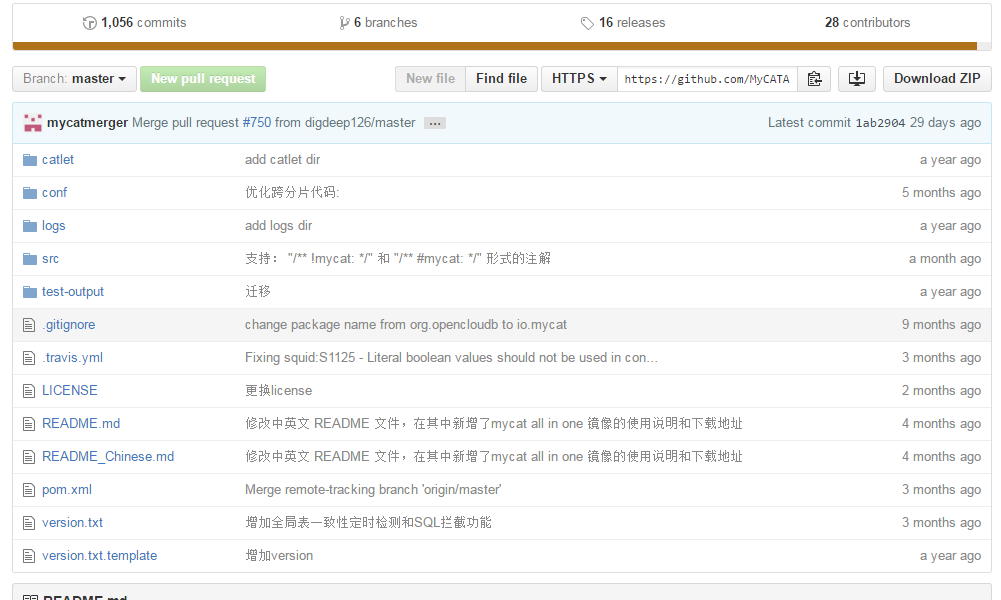
**Mycat 新一代Mysql分布式集群，大数据处理中间件，中国第一开源软件，欢迎志愿者参与：**

**项目源码路径：**[**https://github.com/MyCATApache/Mycat-Server**](https://github.com/MyCATApache/Mycat-Server)



本文档由Mycat志愿者团队提供，转载请注明，谢谢。

## Github使用方法

1. 注册账户以及创建仓库

要想使用github第一步当然是注册github账号了。之后就可以创建仓库了（免费用户只能建公共仓库），Create a New Repository，填好名称后Create，之后会出现一些仓库的配置信息，这也是一个git的简单教程。

1. GitHub协作  
   先从主库fork出自己的分支， 然后进行代码的修改等操作，操作完之后从本地库上推到自己的服务器分支，然后 服务器分支Pull Request到主干。
2. 本地仓库由git维护的三棵“树"组成：第1个是工作目录，它持有实际文件（即可以直接打开修改的文件）；第2个是缓存区（index），它像个缓存区域，临时保存你的改动；最后一个是HEAD，指向你最近一次提交后的结果（这个结果还是在本地电脑里面）。

1）将改动添加到缓存区：

git add <filename>或者 git add \* ;

2)上面是git基本工作流程的第一步，使用如下命令以实际提交改动：

git commit -m "提交的备注信息"

3）现在改动已经提交到了HEAD，但是还没有到你的远程仓库，执行如下将改动提交到远程仓库（可以理解为github网站自己的主页）：其中master可以换成其他想要的分支

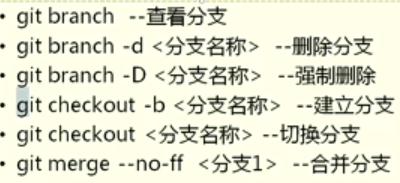
git push origin master

1. 参与社区协作项目的常用命令：



客户端的 git push: 将本地库更新到自己的服务器分支。

1. 管理分支的常用命令：



1. 参与到社区化写作：

  1）进入github的项目的主页面，点击右上角的Fork按钮即可建立自己的分支。（这里fork是指复制到自己的github网站账号上面，要想复制到本地，还需要一些add指令）

    //这里的分支是指上一步fork到自己github网站账号上面的分支

  2）从自己的分支复制到本地。git clone git@github.com:thatcat/manual56.git

   //将主库的URL添加到本地的版本库，然后可以进行下一步的操作

  3）添加主库URL：git remote add mysql2cn git@github.com:mysql2cn/mamual56.git

  4)获取主库的修改：git fetch mysql2cn //将**主库**的修改同步到**本地**

5)合并主库代码到自己的本地分支上：git merge mysql2cn/master master

    //也可以用git pull origin master 同步自己github网站的分支同步到本地

  6)上推到自己的github网站账号：git push origin master

  7)将自己的github网站分支将更新同步到主库：在自己的github网站分支按“pull request”按钮

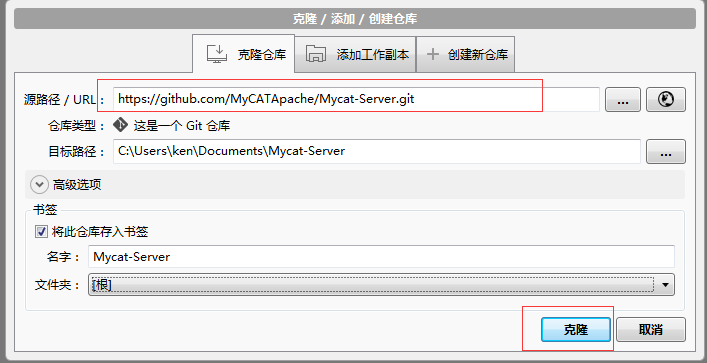
  总结： （1）完整流程：github的项目—>fork到自己的github网站分支—>clone到自己的本地版本—>修改后push到自己的github网站分支—>将自己gitub网站分支pull request到github的原项目

            （2）将原项目的修改同步到自己的分支：github的项目—>fetch、merge到本地版本库—>push到github网站分支—>将自己gitub网站分支pull request到github的原项目

8.创建新仓库：1)在github网站new一个仓库，复制该仓库URL；2）在电脑本地的git bash克隆刚刚new出来的URL:git clone URL；

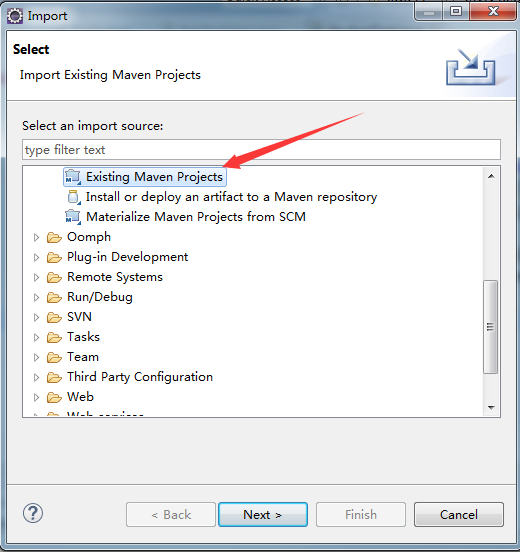
## Checkout项目

采用SourceTree或者Github client来checkout项目

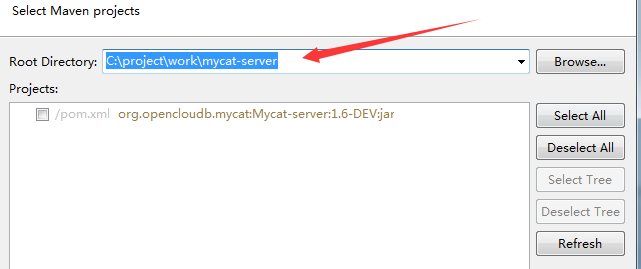


**Eclispe中启动**

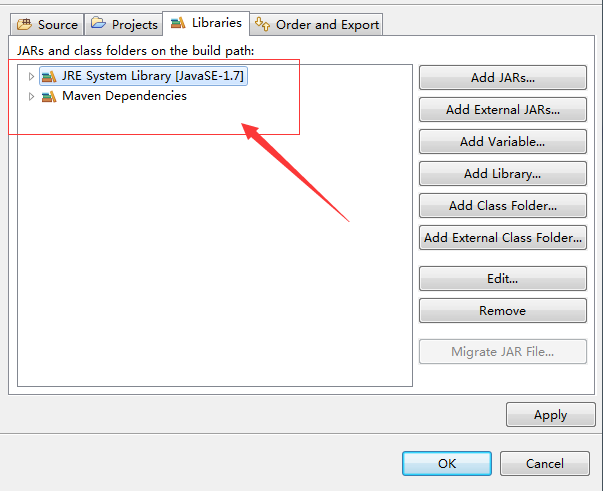
从eclipse Import Mycat-server project ，选择Existing Maven Projects



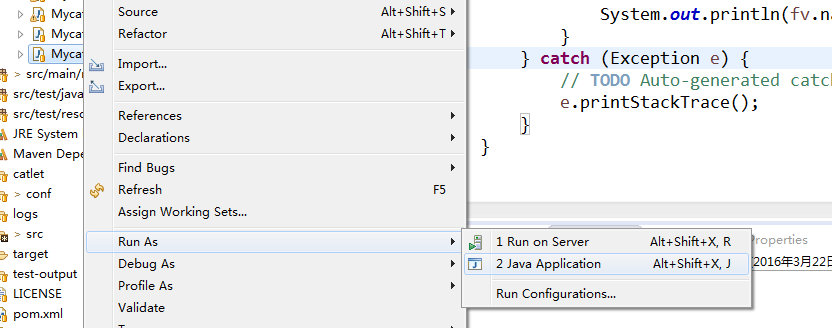
**指定下载完成后的源代码工程路径**



指定JDK版本为1.7



在eclipse中运行MycatStartup 类



## Mycat分片函数的扩展

下面说明分片函数的定义和配置，以Mycat中的例子为例，长整形范围自动分片规则**auto-sharding-long**的定义如下：

<tableRule name="**auto-sharding-long**">

<rule>

<columns>**id**</columns>

<algorithm>**rang-long**</algorithm>

</rule>

</tableRule>

<function name="rang-long" class="org.opencloudb.route.function.AutoPartitionByLong">

<property name="mapFile">autopartition-long.txt</property>

</function>

上述表明：用字段id作为分片字段，并采用命名为**rang-long**的分片函数**： org.opencloudb.route.function.AutoPartitionByLong** 来完成分片映射，此分片函数有一个属性**mapFile**，配置为autopartition-long.txt，意思是从这个文件中加载分片定义：

# range start-end ,data node index

0-2000000=0

2000001-4000000=1

4000001-6000000=2

6000001-8000000=3

8000001-10000000=4

0到2000000的id分片到节点0，2000001到4000000的id分片到节点1，依次类推。

Mycat的分片函数的的目标只有一个：根据某个字段的值，返回其所对应的分片节点DataNode的index，index从零开始；若字段的值为非法，则返回NULL。

用户自定义扩展的分片函数要实现org.opencloudb.config.model.rule.RuleAlgorithm接口：

public interface RuleAlgorithm {

void init();

Integer calculate(String columnValue);

}

分片函数可以定义类的属性，并生成对应的set方法，当此函数被加载的时候，会从 rule.xml文件中找相关的属性，并注入（调用set方法），最后调用init()接口，完成初始化过程，这个过程可以实现资源文件的加载和数据解析。

Calculate方法接受字段的值，并计算返回一个分片节点的index。

附：AutoPartitionByLong的例子

public class AutoPartitionByLong implements RuleAlgorithm {

private String mapFile;

private LongRange[] longRongs;

@Override

public void init() {

initialize();

}

public void setMapFile(String mapFile) {

this.mapFile = mapFile;

}

@Override

public Integer calculate(String columnValue) {

long value = Long.valueOf(columnValue);

Integer rst = null;

for (LongRange longRang : this.longRongs) {

if (value <= longRang.valueEnd && value >= longRang.valueStart) {

return longRang.nodeIndx;

}

}

return rst;

}

private void initialize() {

BufferedReader in = null;

try {

// FileInputStream fin = new FileInputStream(new File(fileMapPath));

InputStream fin = this.getClass().getClassLoader()

.getResourceAsStream(mapFile);

if (fin == null) {

throw new RuntimeException("can't find class resource file "

+ mapFile);

}

in = new BufferedReader(new InputStreamReader(fin));

LinkedList<LongRange> longRangeList = new LinkedList<LongRange>();

for (String line = null; (line = in.readLine()) != null;) {

line = line.trim();

if (line.startsWith("#") || line.startsWith("//"))

continue;

int ind = line.indexOf('=');

if (ind < 0) {

System.out.println(" warn: bad line int "+mapFile+ " :"+line);

continue;

}

try {

String pairs[] = line.substring(0, ind).trim().split("-");

long longStart = Long.parseLong(pairs[0].trim());

long longEnd = Long.parseLong(pairs[1].trim());

int nodeId = Integer.parseInt(line.substring(ind + 1)

.trim());

longRangeList

.add(new LongRange(nodeId, longStart, longEnd));

} catch (Exception e) {

}

}

longRongs = longRangeList.toArray(new LongRange[longRangeList

.size()]);

} catch (Exception e) {

if (e instanceof RuntimeException) {

throw (RuntimeException) e;

} else {

throw new RuntimeException(e);

}

} finally {

try {

in.close();

} catch (Exception e2) {

}

}

}

static class LongRange {

public final int nodeIndx;

public final long valueStart;

public final long valueEnd;

public LongRange(int nodeIndx, long valueStart, long valueEnd) {

super();

this.nodeIndx = nodeIndx;

this.valueStart = valueStart;

this.valueEnd = valueEnd;

}

}

}

单元测试代码如下：

public class AutoPartitionByLongTest {

@Test

public void test()

{

AutoPartitionByLong autoPartition=new AutoPartitionByLong();

autoPartition.setMapFile("autopartition-long.txt");

autoPartition.init();

String idVal="0";

Assert.assertEquals(true, 0==autoPartition.calculate(idVal));

idVal="2000000";

Assert.assertEquals(true, 0==autoPartition.calculate(idVal));

idVal="2000001";

Assert.assertEquals(true, 1==autoPartition.calculate(idVal));

idVal="4000000";

Assert.assertEquals(true, 1==autoPartition.calculate(idVal));

idVal="4000001";

Assert.assertEquals(true, 2==autoPartition.calculate(idVal));

}

}