**13. HBase Java API编程实践**

本实例使用Eclipse编写java程序，来对HBase数据库进行增删改查等操作。

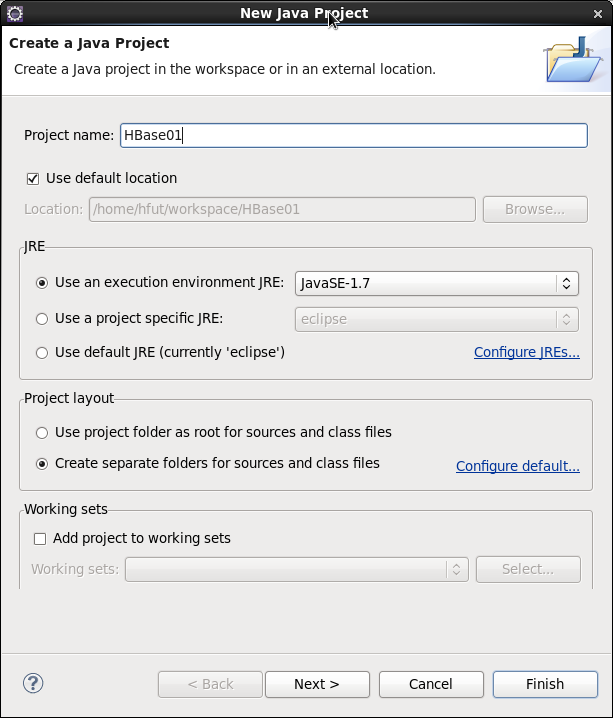
**（1）启动Hadoop和HBase：**

[hfut@master ~]$ start-all.sh

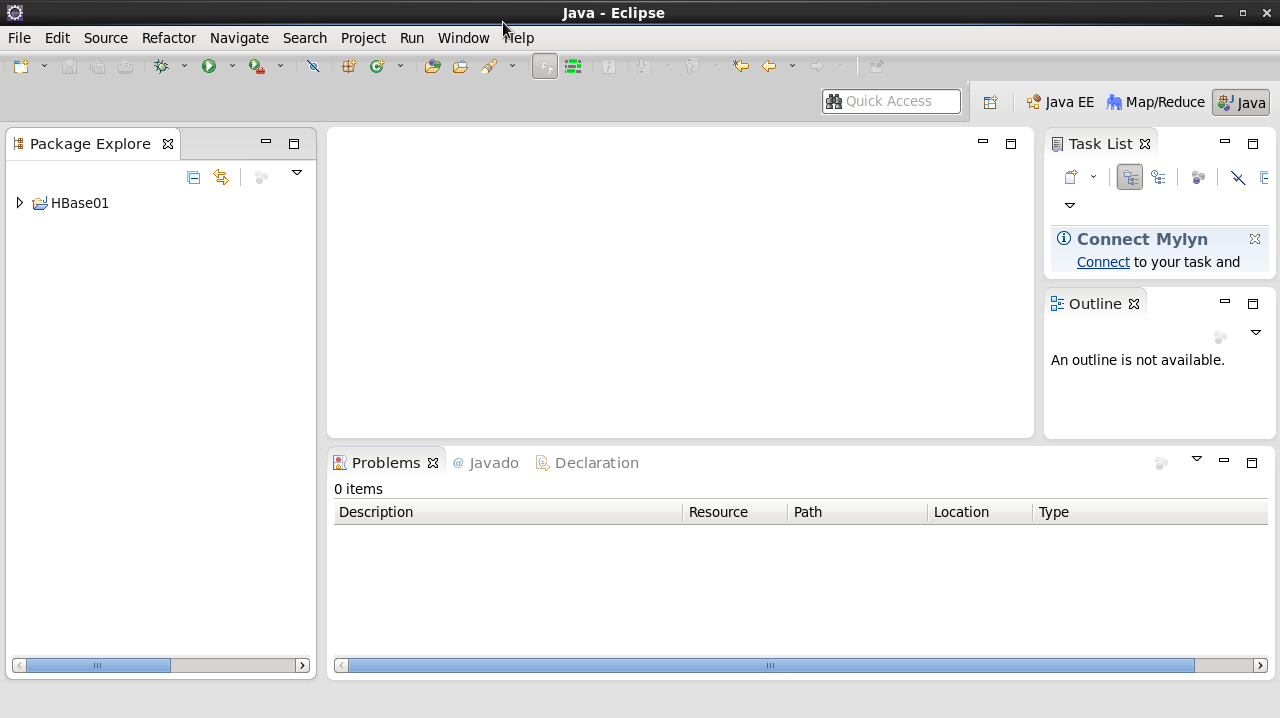
[hfut@master ~]$ start-hbase.sh

**（2）新建Java Project**

新建Java Project：HBase01



点击Finish：

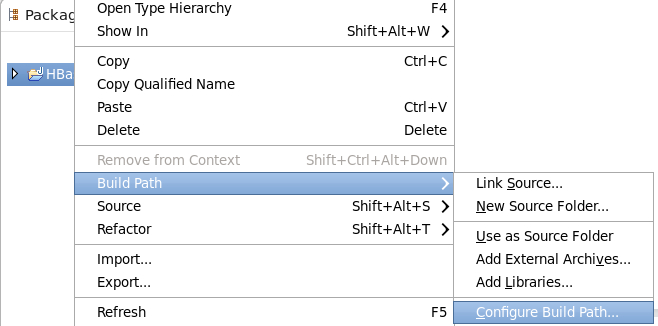


**（3）导入jar包**

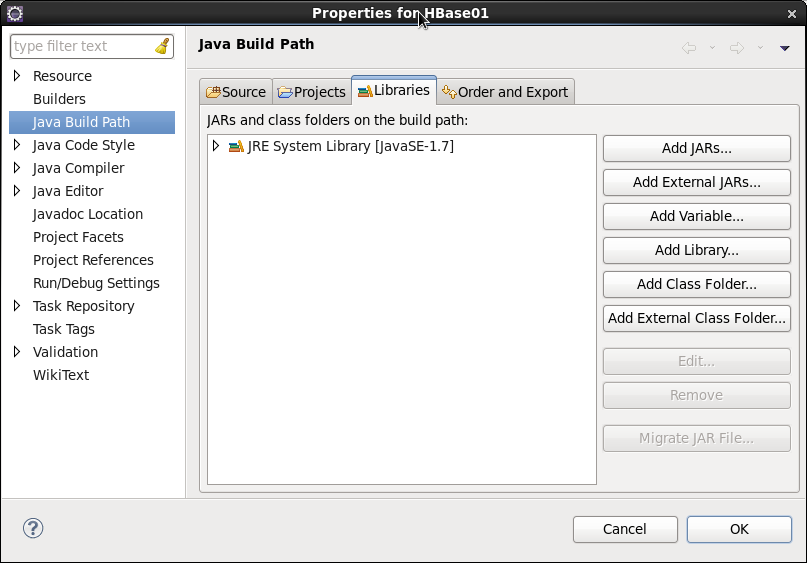
导入hbase安装目录中的lib文件中的所有jar包。

HBase 1.1.2的java api已经发生变化，旧版的部分api已经停止使用，在使用过程中请注意HBase的版本。

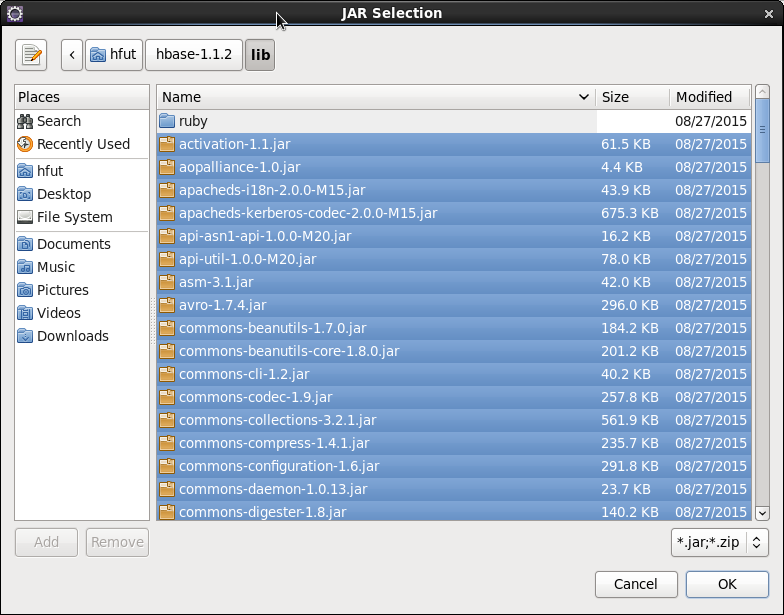
打开HBase01工程的右键菜单：Build Path -> Configure Build Path：



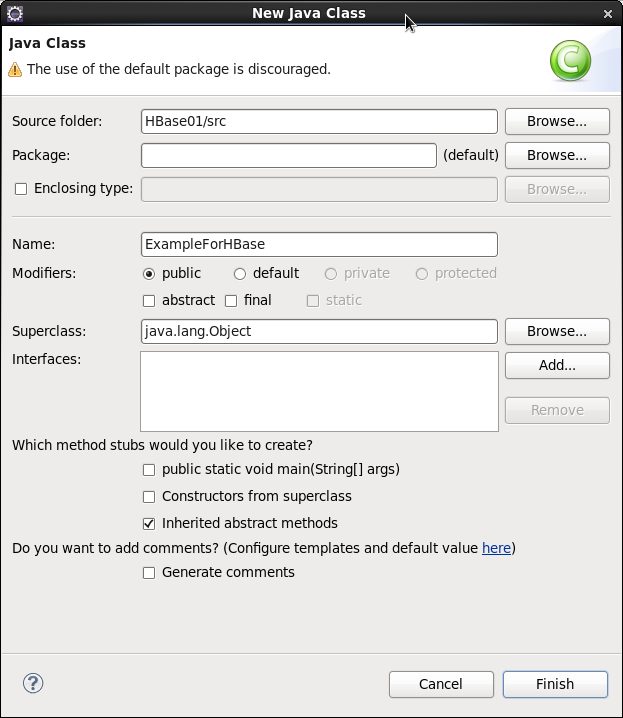
选择Libraries选项卡，点击Add External JARS



全选hbase-1.1.2/lib下的jar文件，点击OK按钮加入。



**（4）新建ExampleForHBase Java Class文件**



并将以下源代码拷贝到ExampleForHBase.java：

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.hbase.\*;

**import** org.apache.hadoop.hbase.client.\*;

**import** java.io.IOException;

**import** java.io.IOException;

**public** **class** ExampleForHBase{

**public** **static** Configuration *configuration*;

**public** **static** Connection *connection*;

**public** **static** Admin *admin*;

//主函数中的语句请逐句执行，只需删除其前的//即可，如：执行insertRow时请将其他语句注释

**public** **static** **void** main(String[] args)**throws** IOException{

//创建一个表，表名为Score，列族为sname,course

//createTable("Score",new String[]{"sname","course"});

//在Score表中插入一条数据，其行键为95001,sname为Mary（因为sname列族下没有子列所以第四个参数为空）

//等价命令：put 'Score','95001','sname','Mary'

//insertRow("Score", "95001", "sname", "", "Mary");

//在Score表中插入一条数据，其行键为95001,course:Math为88（course为列族，Math为course下的子列）

//等价命令：put 'Score','95001','score:Math','88'

//insertRow("Score", "95001", "course", "Math", "88");

//在Score表中插入一条数据，其行键为95001,course:English为85（course为列族，English为course下的子列）

//等价命令：put 'Score','95001','score:English','85'

//insertRow("Score", "95001", "course", "English", "85");

//查询Score表中，行键为95001，列族为course，列为Math的值

//getData("Score", "95001", "course", "Math");

//查询Score表中，行键为95001，列族为sname的值（因为sname列族下没有子列所以第四个参数为空）

//getData("Score", "95001", "sname", "");

//1、删除Score表中指定列数据，其行键为95001,列族为course，列为Math

//执行这句代码前请deleteRow方法的定义中，将删除指定列数据的代码取消注释注释，将删除制定列族的代码注释

//等价命令：delete 'Score','95001','score:Math'

//deleteRow("Score", "95001", "course", "Math");

//2、删除Score表中指定列族数据，其行键为95001,列族为course（95001的Math和English的值都会被删除）

//执行这句代码前请deleteRow方法的定义中，将删除指定列数据的代码注释，将删除制定列族的代码取消注释

//等价命令：delete 'Score','95001','score'

//deleteRow("Score", "95001", "course", "");

//3、删除Score表中指定行数据，其行键为95001

//执行这句代码前请deleteRow方法的定义中，将删除指定列数据的代码注释，以及将删除制定列族的代码注释

//等价命令：deleteall 'Score','95001'

//deleteRow("Score", "95001", "", "");

//删除Score表

//deleteTable("Score");

}

//建立连接

**public** **static** **void** init(){

*configuration* = HBaseConfiguration.*create*();

*configuration*.set("hbase.rootdir","hdfs://master:9000/hbase");

**try**{

*connection* = ConnectionFactory.*createConnection*(*configuration*);

*admin* = *connection*.getAdmin();

}**catch** (IOException e){

e.printStackTrace();

}

}

//关闭连接

**public** **static** **void** close(){

**try**{

**if**(*admin* != **null**){

*admin*.close();

}

**if**(**null** != *connection*){

*connection*.close();

}

}**catch** (IOException e){

e.printStackTrace();

}

}

/\*\*

\* 建表。HBase的表中会有一个系统默认的属性作为主键，主键无需自行创建，默认为put命令操作中表名后第一个数据，因此此处无需创建id列

\* **@param** myTableName 表名

\* **@param** colFamily 列族名

\* **@throws** IOException

\*/

**public** **static** **void** createTable(String myTableName,String[] colFamily) **throws** IOException {

*init*();

TableName tableName = TableName.*valueOf*(myTableName);

**if**(*admin*.tableExists(tableName)){

System.*out*.println("talbe is exists!");

}**else** {

HTableDescriptor hTableDescriptor = **new** HTableDescriptor(tableName);

**for**(String str:colFamily){

HColumnDescriptor hColumnDescriptor = **new** HColumnDescriptor(str);

hTableDescriptor.addFamily(hColumnDescriptor);

}

*admin*.createTable(hTableDescriptor);

System.*out*.println("create table success");

}

*close*();

}

/\*\*

\* 删除指定表

\* **@param** tableName 表名

\* **@throws** IOException

\*/

**public** **static** **void** deleteTable(String tableName) **throws** IOException {

*init*();

TableName tn = TableName.*valueOf*(tableName);

**if** (*admin*.tableExists(tn)) {

*admin*.disableTable(tn);

*admin*.deleteTable(tn);

}

*close*();

}

/\*\*

\* 查看已有表

\* **@throws** IOException

\*/

**public** **static** **void** listTables() **throws** IOException {

*init*();

HTableDescriptor hTableDescriptors[] = *admin*.listTables();

**for**(HTableDescriptor hTableDescriptor :hTableDescriptors){

System.*out*.println(hTableDescriptor.getNameAsString());

}

*close*();

}

/\*\*

\* 向某一行的某一列插入数据

\* **@param** tableName 表名

\* **@param** rowKey 行键

\* **@param** colFamily 列族名

\* **@param** col 列名（如果其列族下没有子列，此参数可为空）

\* **@param** val 值

\* **@throws** IOException

\*/

**public** **static** **void** insertRow(String tableName,String rowKey,String colFamily,String col,String val) **throws** IOException {

*init*();

Table table = *connection*.getTable(TableName.*valueOf*(tableName));

Put put = **new** Put(rowKey.getBytes());

put.addColumn(colFamily.getBytes(), col.getBytes(), val.getBytes());

table.put(put);

table.close();

*close*();

}

/\*\*

\* 删除数据

\* **@param** tableName 表名

\* **@param** rowKey 行键

\* **@param** colFamily 列族名

\* **@param** col 列名

\* **@throws** IOException

\*/

**public** **static** **void** deleteRow(String tableName,String rowKey,String colFamily,String col) **throws** IOException {

*init*();

Table table = *connection*.getTable(TableName.*valueOf*(tableName));

Delete delete = **new** Delete(rowKey.getBytes());

//删除指定列族的所有数据

//delete.addFamily(colFamily.getBytes());

//删除指定列的数据

//delete.addColumn(colFamily.getBytes(), col.getBytes());

table.delete(delete);

table.close();

*close*();

}

/\*\*

\* 根据行键rowkey查找数据

\* **@param** tableName 表名

\* **@param** rowKey 行键

\* **@param** colFamily 列族名

\* **@param** col 列名

\* **@throws** IOException

\*/

**public** **static** **void** getData(String tableName,String rowKey,String colFamily,String col)**throws** IOException{

*init*();

Table table = *connection*.getTable(TableName.*valueOf*(tableName));

Get get = **new** Get(rowKey.getBytes());

get.addColumn(colFamily.getBytes(),col.getBytes());

Result result = table.get(get);

*showCell*(result);

table.close();

*close*();

}

/\*\*

\* 格式化输出

\* **@param** result

\*/

**public** **static** **void** showCell(Result result){

Cell[] cells = result.rawCells();

**for**(Cell cell:cells){

System.*out*.println("RowName:"+**new** String(CellUtil.*cloneRow*(cell))+" ");

System.*out*.println("Timetamp:"+cell.getTimestamp()+" ");

System.*out*.println("column Family:"+**new** String(CellUtil.*cloneFamily*(cell))+" ");

System.*out*.println("ColName:"+**new** String(CellUtil.*cloneQualifier*(cell))+" ");

System.*out*.println("value:"+**new** String(CellUtil.*cloneValue*(cell))+" ");

}

}

}

每次执行完，都可以回到shell界面查看是否执行成功，如：执行完插入数据后，在shell界面中执行scan 'Score'。截图如下：

