# 配表的读取

Workbook workbook = null;  
FileInputStream input = new FileInputStream(file);  
try {  
 workbook = new XSSFWorkbook(input);// 2007 or later  
} catch (Exception e) {  
 workbook = new HSSFWorkbook(input);// 2003 or before  
}

wb.getSheetAt(0);//获取第一张制表

int rows = sheet.getLastRowNum();//得到总行数

T t = clazz.newInstance();//得到Object  
Map<String, String> beanPropMap = BeanUtils.*describe*(t); //对象转Map

// BeanUtils需要两个jar包：commons-beanutils.jar和commons-logging.jar

Map<Integer, String> columnMap = new HashMap<Integer, String>();  
for (int i = 0; i < row.getLastCellNum(); i++) {//得到表头的名字Map  
 Cell cell = row.getCell(i);  
 String name = ExcelUtil.*getCellValue*(cell);  
 if (name != null && beanPropMap.containsKey(name)) {  
 columnMap.put(i, name);  
 }  
}

Map<String, Object> dataMap = new HashMap<String, Object>();  
for (Entry<Integer, String> entry : columnMap.entrySet()) {//获取每行数据  
 int index = entry.getKey();  
 String name = entry.getValue();  
  
 Cell cell = row.getCell(index);  
 if (cell != null) {  
 cell.setCellType(Cell.*CELL\_TYPE\_STRING*);  
 dataMap.put(name, ExcelUtil.*getCellValue*(

String.*valueOf*(cell.getStringCellValue());}  
}

T t = clazz.newInstance();  
BeanUtils.*populate*(t, dataMap);//Map转对象

t.init();

# 游戏缓存的设计。

缓存数据存在GameThreadData下。

private Map<Integer, HashMap<String, BaseCacheObject>> waitUpdateObjData;

含义为：玩家id、缓存数据唯一的key值和缓存的Object。

对于多线程设计了ThreadLocalCache类，并设计单例模式，保证数据的一致。  
public static ThreadLocal<GameThreadData> *localThreadData* = new ThreadLocal<GameThreadData>();

# 协议的传输和映射。

# jfanel的事物增强处理。

# 资源锁。

# 消息队列。

# 数据库的维护。