软件架构设计文档

目录

1 前言

1.1 发布的日期和状态

发布日期：2014-04-04.
状态：进行软件架构的设计。
修改日期：2014-4-24
状态：增加具有接口和数据对象描述

1.2 发布的组织机构

Mosaic Team @ Software Institute of Nanjing University.

1.3 作者

Rhett (hjw12@software.nju.edu.cn).

1.4 变更历史

版　本	作　者	版本描述	日　期
V1.0	Rhett	发布到 SVN 服务器，提供给成员阅读并评审	2014-04-04
V2.0	Rhett	增加了接口和数据对象	2014-4-24

2 总体介绍

2.1 目的

本文档提供 IceBreaker 游戏的软件架构概览，采用若干架构视图描述软件的不同方面，以便表示构造软件所需要的重要架构决策。

2.2 范围

本文档的读者是 Mosaic Team 团队内部的开发和管理人员，参考了[IEEE 1016-2009]中的架构描述推荐文档模版，用于指导下一循环的代码开发和测试工作。

2.3 参考

1. 《软件需求规格说明文档》，Mosaic Team;
2. 《系统设计文档模版》，[IEEE 1016-2009]；
3. 《计算与软件工程（卷三）：团队与软件开发实践》，骆斌，刘嘉，张瑾玉，黄蕾，2012。

2.4 总结

本文档通过软件架构设计过程中的一系列视图勾勒出系统的整个蓝图，为最终代码的开发活动起到指导作用。重在记录系统的软件架构中最重要的信息，作为团队所获取的需求到最终代码之间的映射关系的关键依据。

3 引用

4 词汇表

词 汇 名 称	词 汇 含 义	备 注
IceBreaker	产品游戏的名称	
View	MVC 架构中的视图模块	
Controller	MVC 架构中的控制器模块	
Model	MVC 架构中的模型模块	
Dto	Data transfer object,数据传输对象	封装数据，传输
Dao	Data access object,数据访问对象	直接对数据库进行读写
Entity	数据实体对象	

5 系统设计描述主体

5.1 架构风格

MVC 架构（如图 1）

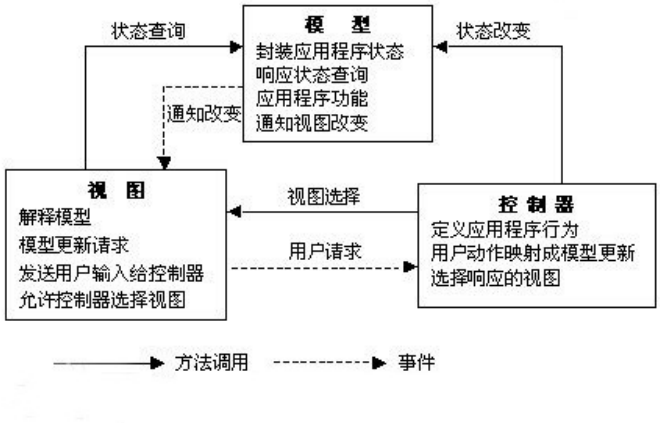

图 1 MVC 组件类型的关系和功能

5.2 系统层次

系统层次结构如图 2 所示。

图 2 系统的层次结构

系统划分为以下 3 个逻辑层次。

1) 展示层：用于前台界面展示和配置的层次。
2) 业务层：包含业务控制和逻辑的层次。
3) 数据层：定义和存储系统中相关数据的层次。

5.3 部署方案

系统部署采用 C/S 架构。系统可以部署在客户端和服务器两个物理层次。

1) 客户端：用户使用软件的层次，展示 UI 和读取用户输入。
2) 服务器端：保存所有用户数据，并为各个用户提供业务逻辑处理，连接多用户的联机操作。

5.4 组件和组件接口

客户端架构中的对象分为 7 类：

1) View 对象，负责 UI 的展示和与用户的交互。
2) Viewservice 对象，模型模块改变视图调用的服务。
3) Controller 对象，负责获取用户的输入和改变模型。
4) Modelservice 对象，控制器或服务器反馈信息改变模型调用的服务。
5) Model 对象，封装游戏信息，响应视图的查询和控制器的改变。
6) Netservice 对象，模型模块向服务器进行请求操作调用的服务。
7) Net 对象，负责网络服务接口的实现模块，与服务器进行连接。

客户端的组件和组件接口：

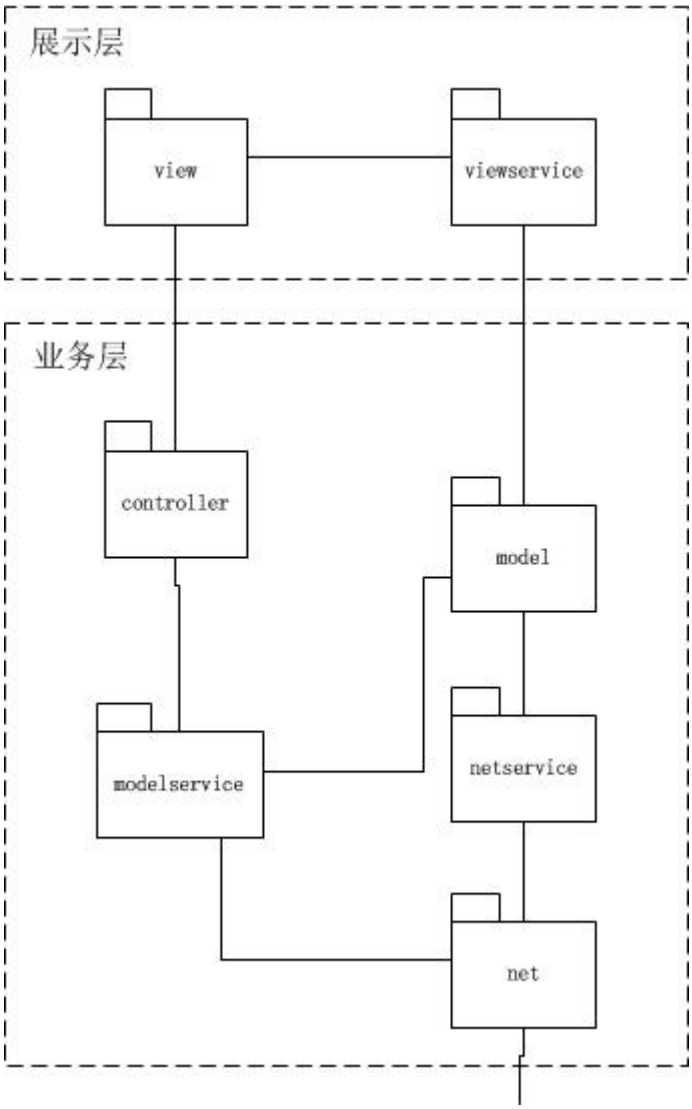

图 3 客户端的组件

接口 ID	连接组件	接口信息	
I1	连接 view 和 controller	语法	Return(Response) Interface(Request)
		前置条件	用户的输入正确
		后置条件	经过控制组件处理请求并且响应
		不变量	用户请求信息
I2	连接 controller 和 modelservice	语法	Return() Interface(Request)
		前置条件	无
		后置条件	模型经过逻辑处理进行相应改变
		不变量	无
I3	连接 model 和 viewservice	语法	Return() Interface()
		前置条件	无
		后置条件	视图响应模型进行改变
		不变量	无
I4	连接 model 和 netservice	语法	Return() Interface(Request)
		前置条件	与服务器正常连接
		后置条件	服务器收到请求进行处理
		不变量	模型状态
I5	连接 net 和 modelservice	语法	Return() Interface()
		前置条件	无
		后置条件	模型收到服务器反馈结果进行相应改变
		不变量	无

服务器架构中的对象分为 6 类：

1) Net 对象，负责与客户端的连接。
2) Service 对象，调用服务器逻辑处理的服务。
3) Serviceimp 对象，负责对于抽象接口的服务模块。
4) Dto 对象，负责封装从 dao 获取的批量数据的接口。
5) Dao 对象，负责与数据库实体交互，获取数据。
6) Entity 对象，负责将数据库中获取的数据封装成数据尸体。

服务器端的组件和组件接口：

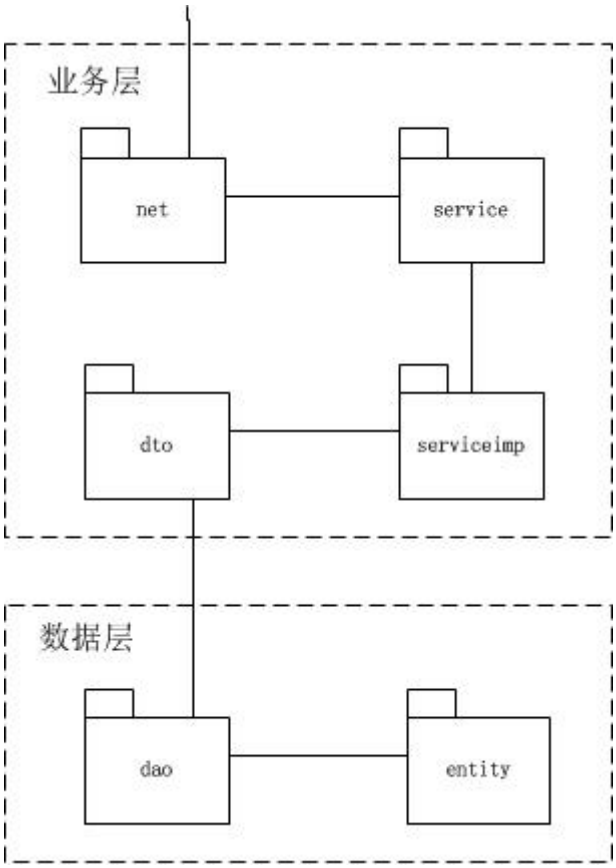

图 4 服务器端的组件

接口 ID	连接组件	接口信息	
I1	连接 net 和 service	语法	Return(result) Interface()
		前置条件	无
		后置条件	Service 进行逻辑处理并返回结果
		不变量	无
I2	连接 service 和 dto	语法	Return(dataSet) Interface(command)
		前置条件	无
		后置条件	对应的 dto 组件调用特定 dao 类获取数据层数据，并返回数据集
		不变量	无
I3	连接 dao 和 entity	语法	Return(data) Interface(criteria)
		前置条件	数据库连接正常
		后置条件	Dao 中的类将 entity 对象写入数据库或从数据库中返回 entity 对象
		不变量	无

5.5 业务逻辑层模块接口规范

这里描述主要的模型（model）模块接口

表 5-5-1 BoardModel 模块接口规范

提供的服务（供接口）
BoardModel.trySwap
BoardModel.useToolGrid
需要的服务（需接口）
BoardView.initBoard
BoardView.swapUnit
BoardView.deleteUnits
BoardView.dealWithMoveAction
NetService.act

表 5-5-2 HelpModel 模块接口规范

提供的服务（供接口）
HelpModel.fetchHelpTips
需要的服务（需接口）
无

表 5-5-3 InfoModel 模块接口规范

提供的服务（供接口）
InfoModel.fetchInfo
InfoModel.ifBought
需要的服务（需接口）
InfoView.setCharacter
InfoView.showInfo
NetService.getInfo

表 5-5-4 MenuModel 模块接口规范

提供的服务（供接口）
MenuModel.friendRank
MenuModel.worldRank
需要的服务（需接口）
MainView.refreshFriendsRank
MainView.refreshAllRank
NetService.getMyRank
NetService.getWorldRank

表 5-5-5 ReadyModel 模块接口规范

提供的服务（供接口）
ReadyModel.askCouple

ReadyModel.getFriends
需要的服务（需接口）
GameView.ToWaitFoAgree
GameView.ToGame
NetService.getOnlinePal
NetService.cooperateWithPal
NetService.cooperateRandom
NetService.pkWithPal
NetService.pkRandom

表 5-5-6 RegisterModel 模块接口规范

提供的服务（供接口）
RegisterModel.Register
需要的服务（需接口）
NetService.register

表 5-5-7 SetModel 模块接口规范

提供的服务（供接口）
SetModel.setMusicVolume
SetModel.setSoundVolume
需要的服务（需接口）
无

表 5-5-8 ToolModel 模块接口规范

提供的服务（供接口）
ToolModel.buyTool
ToolModel.toolNumber
需要的服务（需接口）
ToolView.updateToolNumbers
NetService.buyTool
NetService.getMyTools

表 5-5-9 UserModel 模块接口规范

提供的服务（供接口）
UserModel.login
UserModel.responseLogin
需要的服务（需接口）
LoginView.dealWithLogin
NetService.signIn

表 5-5-10 ClientModel 模块接口规范

提供的服务（供接口）
ClientModel.login

ClientModel.register
ClientModel.myInfo
ClientModel.myTools
ClientModel.newRecord
ClientModel.levelUp
ClientModel.invite
ClientModel.PK
ClientModel.startCooperate
ClientModel.startPK
ClientModel.remove
ClientModel.removeTogether
ClientModel.ranks
需要的服务（需接口）
无

6.信息视角

6.1 数据持久化对象

系统的 VO 类就是对应的相关的实体类,是抽象出来的业务对象

●AccountVO 类包含用户的用户名、密码属性

●GainVO 类包含游戏的成就

●InfoVO 类包含用户的个人信息

●MoveandCreateActionVO 类包含游戏中对盘面的操作动作

●PositionVO 类包含游戏中盘面的位置

●ReplyVO 类包含双人/pk 模式中对配对请求的响应

●ScoreVO 类包含游戏中的得分

●ToolVO 类包含游戏中的道具

持久化用户对象 AccountVO 的定义如图 6-1 所示:

```
public class AccountVO implements Serializable{

   private static final long serialVersionUID = 8904862036172036545L;
   private String id;
   private String pwd;

   public AccountVO(String id,String pwd){
      this.id = id;
      this.pwd = pwd;
```

```
    }

    public String getID(){
        return id;
    }

    public String getPassword(){
        return pwd;
    }

}
```

图 6-1 持久化用户对象 AccountVO 的定义

持久化用户对象 GainVO 的定义如图 6-2 所示:

```
public class AccountVO implements Serializable{

    private static final long serialVersionUID = 8904862036172036545L;
    private String id;
    private String pwd;

    public AccountVO(String id,String pwd){
        this.id = id;
        this.pwd = pwd;
    }

    public String getID(){
        return id;
    }

    public String getPassword(){
        return pwd;
    }

}
```

图 6-2 持久化用户对象 GainVO 的定义

持久化用户对象 InfoVO 的定义如图 6-3 所示:

```
public class InfoVO implements Serializable{

    /**
```

```
     *
     */
    private static final long serialVersionUID = -1525398331705977472L;
    private String id;//id
    private int level;//等级
    private int exp;//经验
    private int best;//最好成绩
    private int coin;//金币

    public InfoVO(String id,int level,int exp,int best,int coin){
        this.id = id;
        this.level = level;
        this.exp = exp;
        this.best = best;
        this.coin = coin;
    }

    public String getID(){
        return id;
    }

    public int getLevel(){
        return level;
    }

    public int getExp(){
        return exp;
    }

    public int getBest(){
        return best;
    }

    public int getCoin(){
        return coin;
    }

}
```

图 6-3 持久化用户对象 InfoVO 的定义

持久化用户对象 MessageVO 的定义如图 6-4 所示:

```
public class MessageVO implements Serializable{
```

```
    private static final long serialVersionUID = -3251310226508238364L;
    private String cmd;
    private Object obj;

    public MessageVO(String cmd,Object obj){
        this.cmd = cmd;
        this.obj = obj;
    }

    public String getCommand(){
        return cmd;
    }

    public Object getObject(){
        return obj;
    }

}
```

图 6-4 持久化用户对象 MessageVO 的定义

持久化用户对象 MoveandCreateActionVO 的定义如图 6-5 所示:

```
public class MoveandCreateActionVO{
    //0代表是盘面已有的方块进行移动，1代表是新方块生成
    private int type;

    //方块移动的起始位置，如果是生成新方块，位置是动画效果中的界外的位置
    private PositionVO gridOld;

    //颜色，仅适用于生成新方块
    private int color;

    //移动方向，0为向下，1为向右
    private int direction;

    //移动的格数
    private int steps;

    //moveaction的构造
    public MoveandCreateActionVO(int type,PositionVO gridOld,int
direction,int steps) throws Exception{
        if(type!=0){
            throw new Exception("类型与构造函数不匹配");
```

```
        }
        this.type = 0;
        this.gridOld = gridOld;
        this.direction = direction;
        this.steps = steps;

    }

    //createaction的构造
    public MoveandCreateActionVO(int type,int color,PositionVO gridOld,int
direction,int steps) throws Exception{
        if(type!=1){
            throw new Exception("类型与构造函数不匹配");
        }
        this.type = 1;
        this.color = color;
        this.gridOld = gridOld;
        this.direction = direction;
        this.steps = steps;
    }

    public int type(){
        return type;
    }

    public int color() throws Exception{
        if(type!=1){
            throw new Exception("类型与函数不匹配");
        }
        return color;
    }

    public PositionVO gridOld(){
        return gridOld;
    }

    public int direction(){
        return direction;
    }

    public int steps(){
        return steps;
    }
}
```

图 6-5 持久化用户对象 MoveandCreateActionVO 的定义

持久化用户对象 PositionVO 的定义如图 6-6 所示：

```
public class PositionVO {
   //网格中的位置 x,y都从0开始
   //网格外的位置？？
   private int x;
   private int y;

   public PositionVO(int x,int y){
      this.x = x;
      this.y = y;
   }

   public boolean adjacent(PositionVO another){
      int x2 = another.getX();
      int y2 = another.getY();
      int tmp = Math.abs(x-x2)+Math.abs(y-y2);
      return (tmp==1);
   }

   public int getX(){
      return x;
   }

   public int getY(){
      return y;
   }

   public boolean isValid(PositionVO v2){

   if(((this.getX()-v2.getX()==0)&&Math.abs(this.getY()-v2.getY())==1)||
((this.getY()-v2.getY()==0)&&Math.abs(this.getX()-v2.getX())==1))
         return true;
      else
         return false;
   }
}
```

图 6-6 持久化用户对象 PositionVO 的定义

持久化用户对象 ScoreVO 的定义如图 6-7 所示：

```
public class ScoreVO implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = -7083295618997381308L;
    private String id;
    private int score;

    public ScoreVO(String id,int score){
        this.id = id;
        this.score = score;
    }

    public String getID(){
        return id;
    }

    public int getScore(){
        return score;
    }

}
```

图 6-7 持久化用户对象 ScoreVO 的定义

持久化用户对象 ReplyVO 的定义如图 6-8 所示:

```
/**
     *
     */
    private static final long serialVersionUID = 6230384551054621879L;
    private boolean isAgree;// 是否同意进行协作或pk
    private String palID;// 对方id

    public ReplyVO(boolean i, String p) {
        isAgree = i;
        palID = p;
    }

    public boolean getAgreement() {
        return isAgree;
    }

    public String getPalID() {
```

```
        return palID;
    }

}
```

图 6-8 持久化用户对象 ReplyVO 的定义

持久化用户对象 ToolVO 的定义如图 6-9 所示:

```
public class ToolVO implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 5314183648771894371L;
    private String tool_id;//道具id
    private int num;//道具数量

    public ToolVO(String tool_id,int num){
        this.tool_id = tool_id;
        this.num = num;
    }

    public String getToolID(){
        return tool_id;
    }

    public int getNum(){
        return num;
    }

}
```

图 6-9 持久化用户对象 ToolVO 的定义