**软件架构设计文档**

目录

[1 前言 1](#_Toc386401455)

[1.1 发布的日期和状态 1](#_Toc386401456)

[1.2 发布的组织机构 1](#_Toc386401457)

[1.3 作者 1](#_Toc386401458)

[1.4 变更历史 1](#_Toc386401459)

[2 总体介绍 1](#_Toc386401460)

[2.1 目的 1](#_Toc386401461)

[2.2 范围 1](#_Toc386401462)

[2.3 参考 2](#_Toc386401463)

[2.4 总结 2](#_Toc386401464)

[3 引用 2](#_Toc386401465)

[4 词汇表 2](#_Toc386401466)

[5 系统设计描述主体 2](#_Toc386401467)

[5.1 架构风格 2](#_Toc386401468)

[5.2 系统层次 3](#_Toc386401469)

[5.3 部署方案 3](#_Toc386401470)

[5.4 组件和组件接口 4](#_Toc386401471)

[5.5 业务逻辑层模块接口规范 7](#_Toc386401472)

[6.信息视角 9](#_Toc386401473)

[6.1 数据持久化对象 9](#_Toc386401474)

# 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](mailto: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）

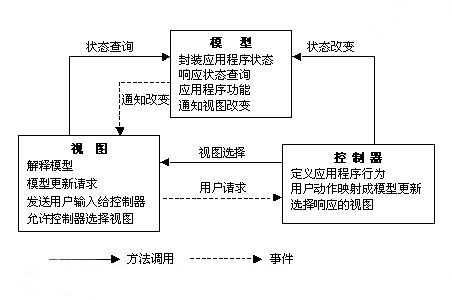


图1 MVC组件类型的关系和功能

## 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：

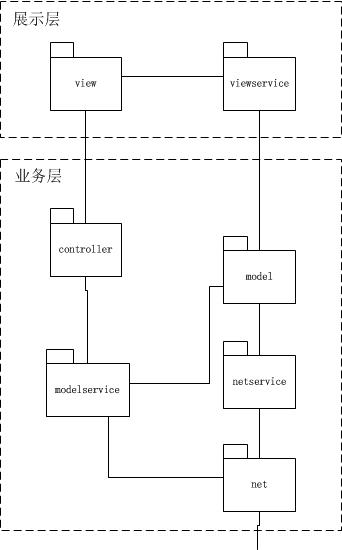


图3 客户端的组件

|  |  |  |  |
| --- | --- | --- | --- |
| **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：

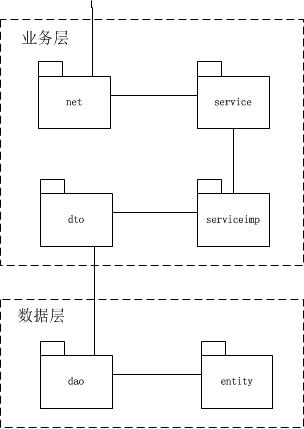


图4 服务器端的组件

|  |  |  |  |
| --- | --- | --- | --- |
| **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