**NBA数据查询系统**

**项目设计文档**

Daisy小组

张孙暮雨(131250215)(PM)

周子懿(131250176)

王宁(131250220)

倪爽(131250211)

**2015/3/21**

[1.引言 4](#_Toc11724)

[1.1编写目的 4](#_Toc20534)

[1.2对象与范围 4](#_Toc18394)

[1.3参考文献 4](#_Toc5496)

[1.4名词与术语 4](#_Toc23412)

[2.系统分层架构 5](#_Toc22441)

[3.逻辑视角 6](#_Toc10360)

[4.组合视角 8](#_Toc5616)

[4.1开发包图 8](#_Toc23882)

[4.2运行时进程 10](#_Toc2475)

[4.3物理部署 11](#_Toc10284)

[4.4构件图 11](#_Toc8423)

[5.接口视角 13](#_Toc15468)

[5.1模块的职责 13](#_Toc4459)

[5.2用户界面层的分解 15](#_Toc12728)

[5.2.1用户界面层模块的职责 16](#_Toc28963)

[5.2.2用户界面层模块的接口规范 16](#_Toc5979)

[5.2.3用户界面模块设计原理 17](#_Toc26361)

[5.3业务逻层的分解 17](#_Toc8761)

[5.3.1业务逻辑层模块的职责 18](#_Toc24412)

[5.3.2业务逻辑层模块的接口规范 18](#_Toc21434)

[5.4数据层的分解 20](#_Toc18941)

[5.4.1数据层模块的职责 21](#_Toc10506)

[5.4.2数据层模块的接口规范 21](#_Toc21141)

[5.5模块的关键类图 22](#_Toc7507)

[6．信息视角 25](#_Toc21673)

[6.1数据持久化对象 25](#_Toc24292)

[6.2 database数据表格式 25](#_Toc2732)

**更新历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改人员 | 日期 | 变更原因 | 版本号 |
| 倪爽 | 2015/3/7 | 最初草稿 | V1.0草稿 |
| 周子懿 | 2015/3/18 | 添加数据层接口规范并完善文档 | V2.0 |
| 王宁 | 2015/3/21 | 添加逻辑层接口规范和关键类图 | V2.1 |

# 1.引言

## 1.1编写目的

本文档提供了NBA数据查询软件的架构概览。采用若干架构视图描述系统的不同方面，以便表示构造系统所需要的重要架构决策。

## 1.2对象与范围

本文档的读者是Daisy团队内部的开发和管理人员，用于指导下一循环的代码开发和测试工作。

## 1.3参考文献

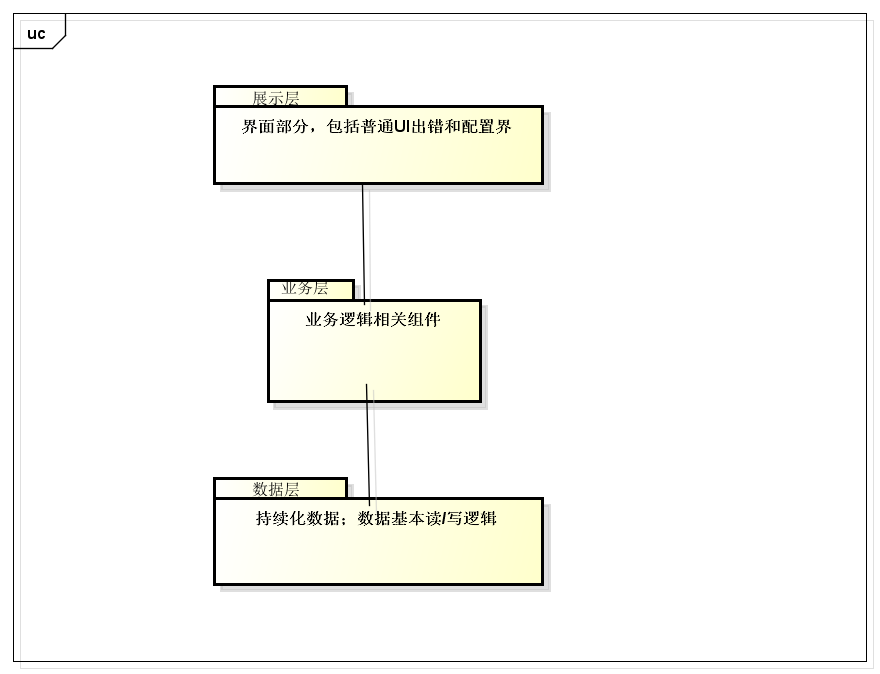
1.《NBA数据查询系统规格说明书V2.0》，Daisy团队

2.《软件架构文档模板》

**1.4名词与术语**

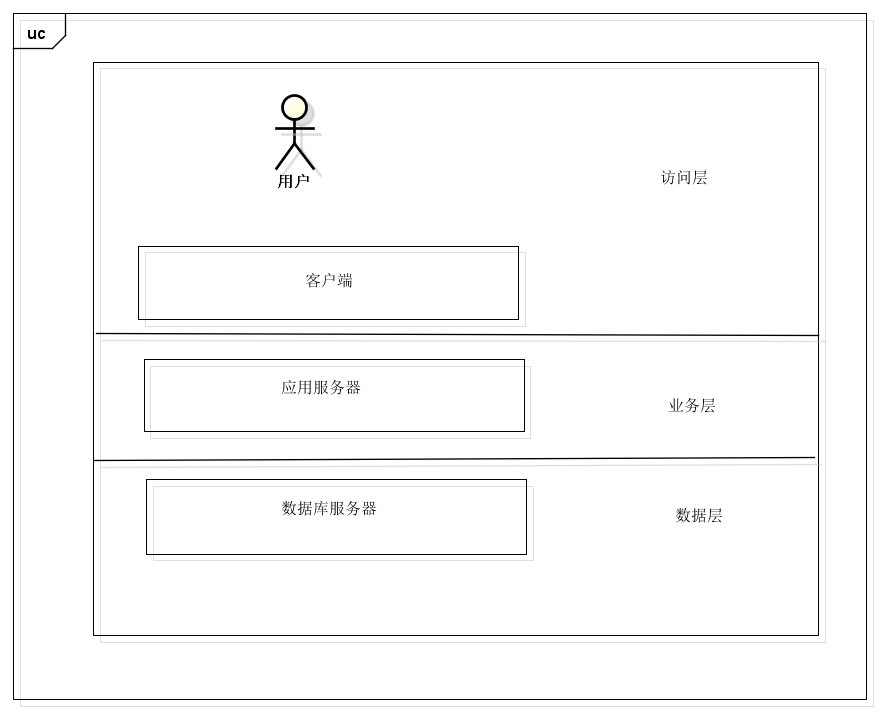
|  |  |
| --- | --- |
| **名词** | **释义** |
| NBA | 美国及加拿大职业篮球联盟 |

# 2.系统分层架构



系统划分为以下4个逻辑层次。

1. 表示层：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。



系统可以部署在以下3个物理层次。

（1）访问层：用于用户访问系统的层次。

（2）业务层：部署业务控制和逻辑的层次。

（3）数据层：部署和存储系统中相关数据的层次。

# 3.逻辑视角

NBA数据查询系统中，选择了分层体系结构风格，将系统分为3层（展示层、业务逻辑层、数据层）能够很好地示意整个高层抽象。展示层包含GUI页面的实现，业务逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图3-1和如图3-2所示。



图3-1 参照体系结构风格的包图表达逻辑视角

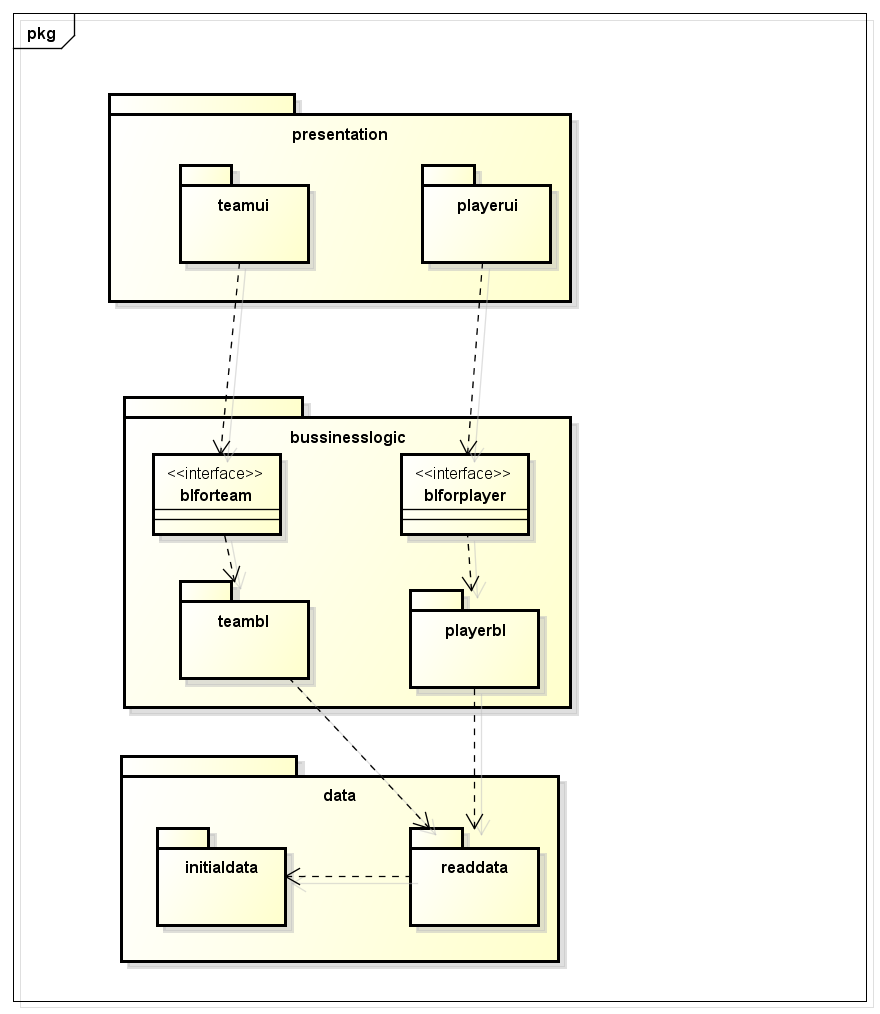


图3-2 软件体系结构逻辑设计方案

# 4.组合视角

## 4.1开发包图

NBA数据查询系统的最终开发包设计如表4.1-1

表4.1-1 NBA数据查询系统的最终开发包设计

|  |  |
| --- | --- |
| 开发包（物理） | 依赖的其他开发包 |
| mainui | playerui, teamui |
| playerui | playerblservice, 界面类库包, vo |
| playerblservice |  |
| playerbl | playerblservice, playerdataservice,po |
| playerdataservice | javaRMI,po |
| playerdata | po, playerdataservice |
| teamui | teamblservice, 界面类库包, vo |
| teamblservice |  |
| teambl | teamblservice, teamdataservice,po |
| teamdataservice | javaRMI,po |
| teamdata | po, teamdataservice |
| vo |  |
| po |  |
| 界面类库包 | Java.swing,java.awt,java.2D,java.3D |
| javaRMI |  |
| txtutility |  |

NBA数据查询系统系统客户端开发包图如图4.1-1所示,服务器端开发包图如图4.1-2所示

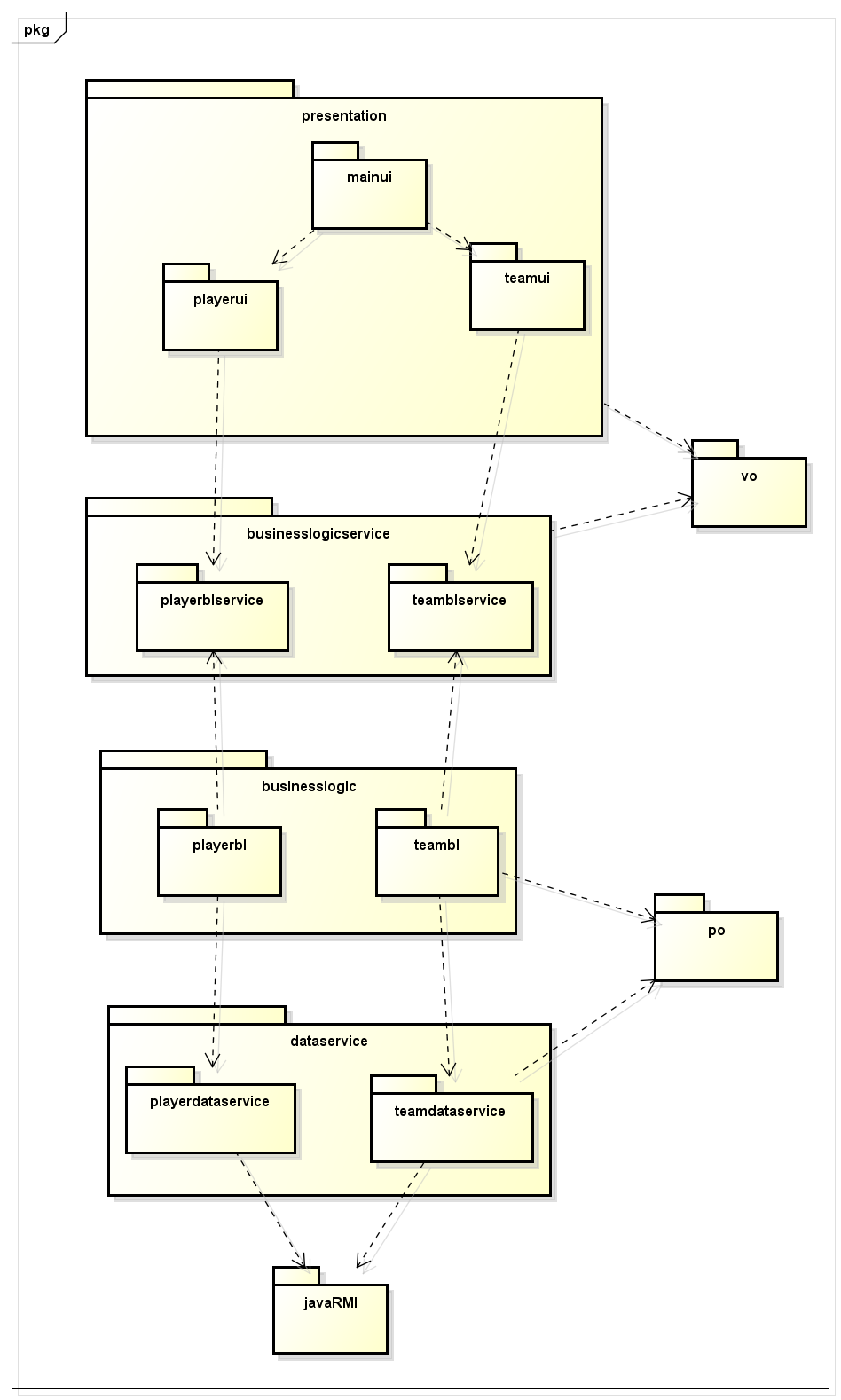


图4.1-1 NBA数据查询系统客户端开发包图

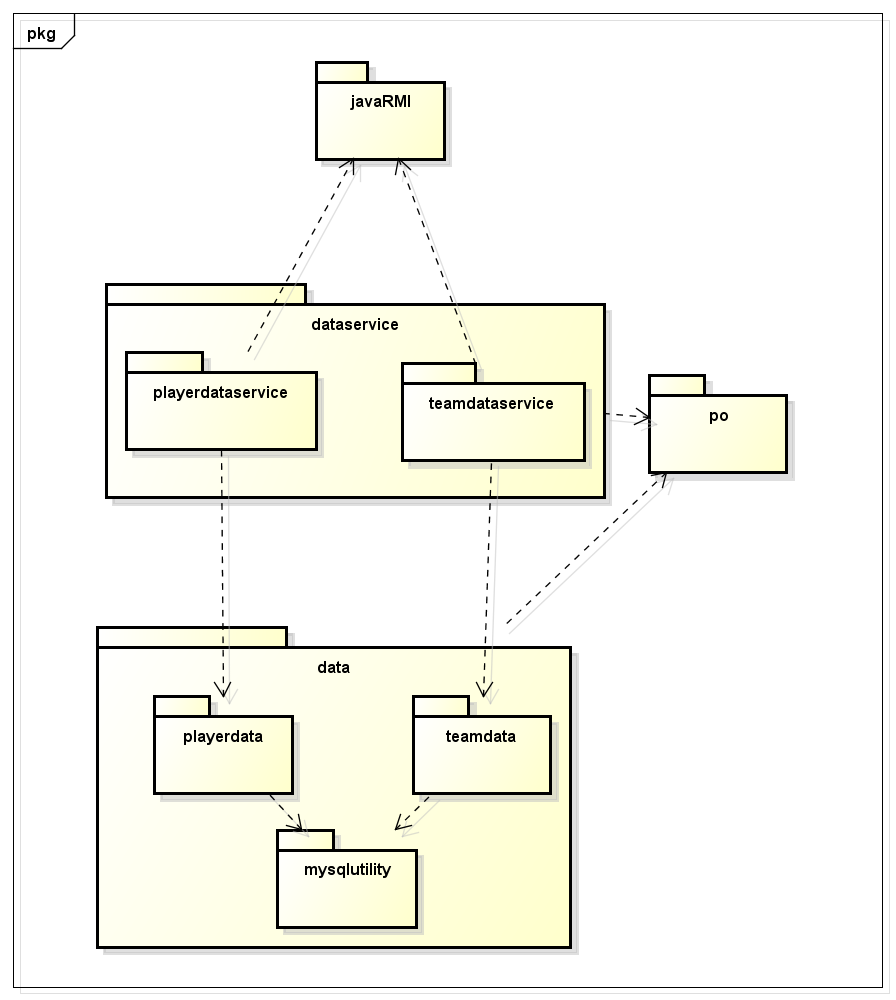


图4.1-2 NBA数据查询系统服务器端开发包图

## 4.2运行时进程

在NBA数据查询系统中,会有多个客户端进程和一个服务器端进程,其进程图如图4.2所示。结合部署图,客户端进程是在客户端机器上运行,服务器端进程在服务器端机器上运行。



图4.2 进程图

## 4.3物理部署

NBA数据查询系统中客户端构件是放在客户端机器上，服务器端构件是放在服务器端机器上。在客户端节点上，还要部署RMIStub构件。由于JavaRMI构件属于JDK6.0的一部分。所以，在系统JDK环境已经设置好的情况下，不需要再独立部署。部署图如图4.3所示。

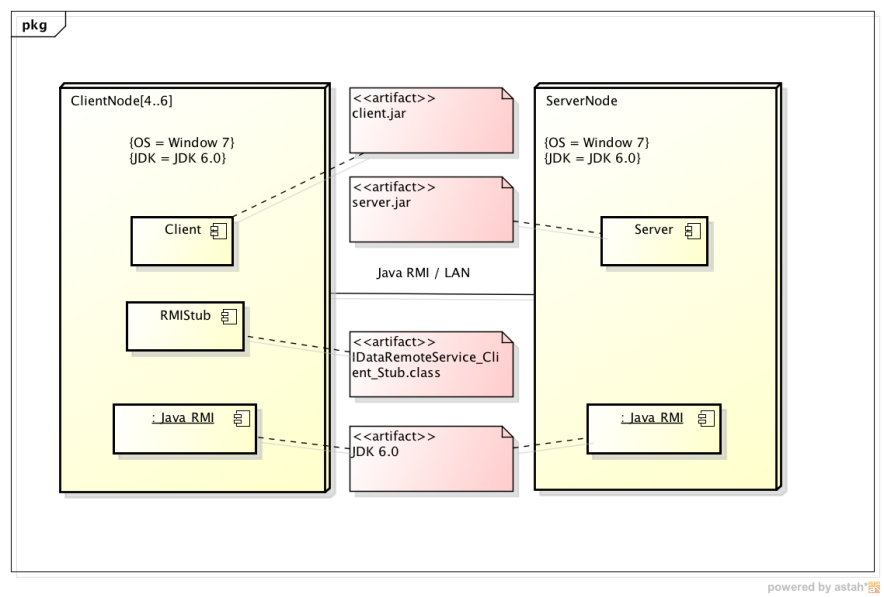


图4.3 部署图

## 4.4构件图

NBA数据查询系统有3个顶级构件：客户端、服务器端、以及负责连接两端的网络构件。从体系结构的角度，客户端和服务器端是构件（Component）；而网络构件是连接件（Connector）。每个顶层构件还可以进一步再细分出部分。

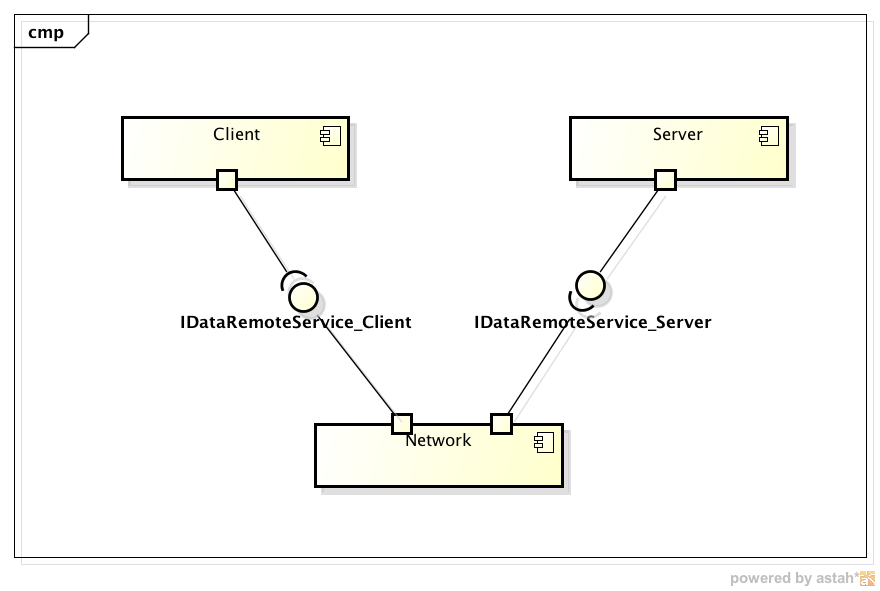


图4.4-1 0层构件图

1层构件图主要进一步细化了0层构件。这时候，需要考虑到逻辑视角的映射，通过1层构件图将逻辑视角的分层映射到0层构件图中的构件中去。逻辑视角的各层可能会分别部署在不同的物理机器上。

NBA数据查询系统，数据层是放在服务器端的。客户端和服务器端都有各自的展示层和业务逻辑层。不同的是，客户端需要通过网络构件连接数据层的IDataRemoteService接口，而服务器端则是业务逻辑层直接访问IDataService接口。对于Network构建，我们也可以细化。由于使用了RMI技术，所以客户端需要有Stub类。Stub类和服务器端的需接口都共同依赖与Java RMI构件。

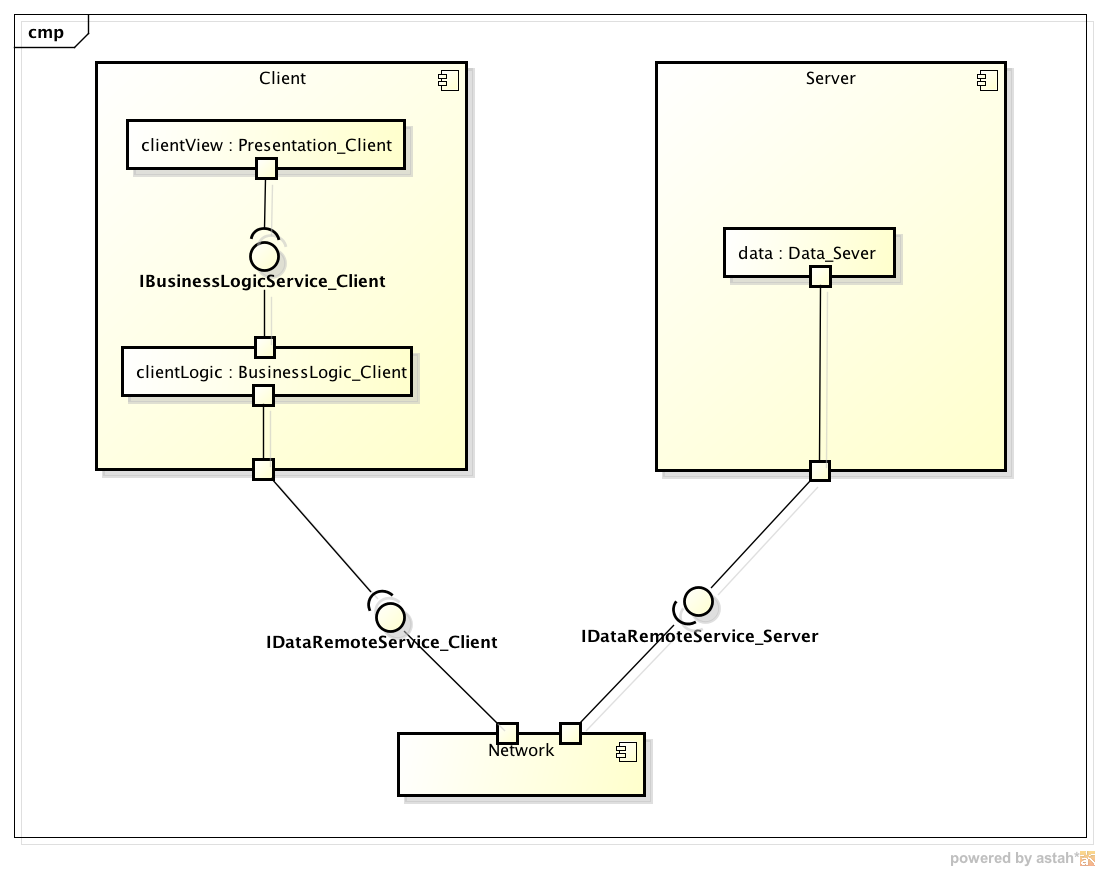


图4.4-2 1层构件图

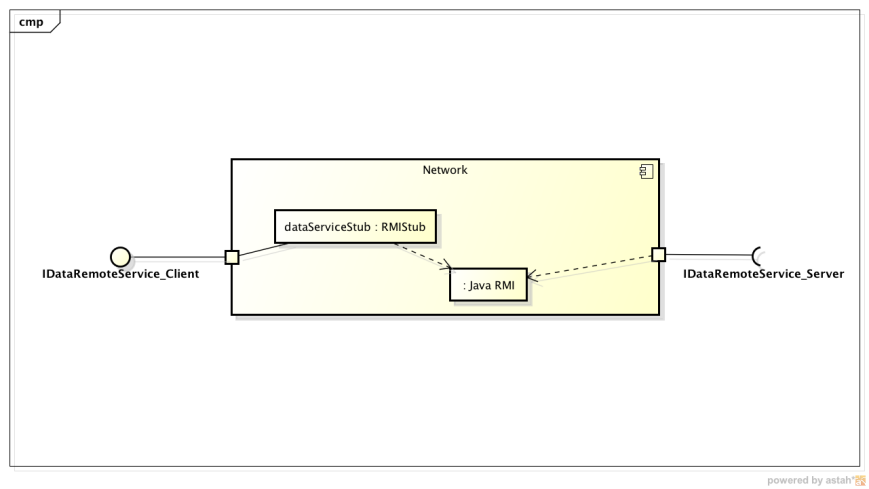


图4.4-3 Network构件图

# 5.接口视角

## 5.1模块的职责

客户端模块和服务器端模块试图分别如图5.1-1和图5.1-2所示。客户端各层和服务器端各层的职责分别如表5.1-1和表5.1-2所示。



图5.1-1 客户端模块视图



图5.1-2 服务器端模块视图

表5.1-1 客户端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面。 |
| 用户界面层 | 基于窗口的NBA数据查询系统客户端用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

表5.1-2 服务器端各层的职责

|  |  |
| --- | --- |
| 层 | 职责 |
| 启动模块 | 负责初始化网络通信机制，启动用户界面。 |
| 用户界面层 | 基于窗口的NBA数据查询系统服务器用户界面。 |
| 业务逻辑层 | 对于用户界面的输入响应和业务处理逻辑。 |
| 数据层 | 负责数据的持久化及数据访问接口。 |
| 服务器端网络模块 | 利用Java RMI机制开启RMI服务，注册RMI服务 |

表5.1-3 层之间调用的接口

|  |  |  |
| --- | --- | --- |
| 接口 | 服务调用方 | 服务提供方 |
| playerblservice  teamblservice | 客户端展示层 | 客户端业务逻辑层 |
| playerdataservice  teamdataservice | 客户端业务逻辑层 |  |

借用球员查看用例来说明层之间的调用，如图5.1-3所示。每一层之间都是由上层依赖了一个接口（需接口），而下层实现这个接口（供接口）。playerblservice提供了框架界面所需要的所有业务逻辑功能。playerdataservice提供了对数据库的增、删、改、查等操作。这样的实现就大大降低了层与层之间的耦合。

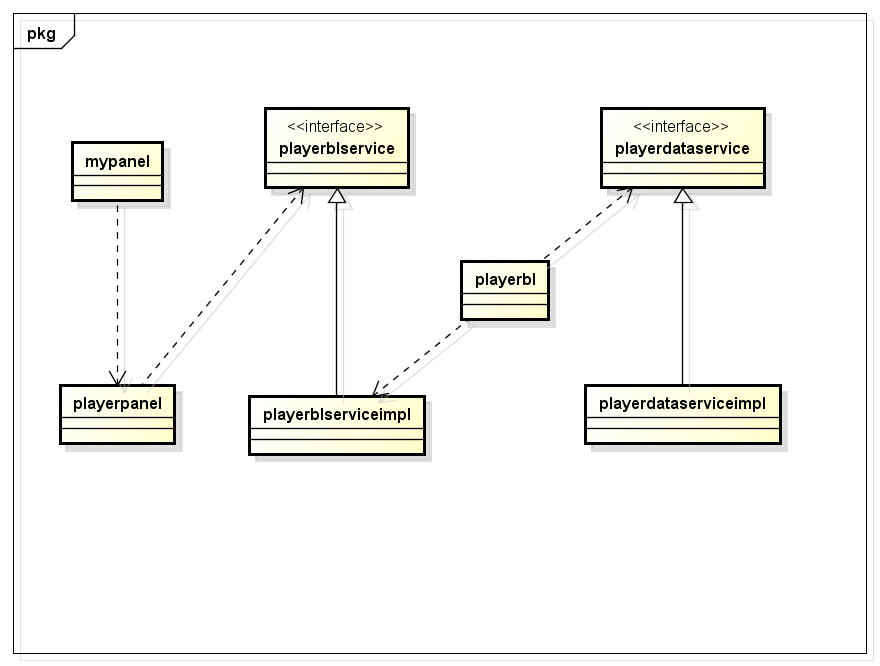


图5.1-3 球员查看用例层之间调用的接口

## 5.2用户界面层的分解

根据需求，系统存在24个用户界面：主界面、球队排名界面、球队信息显示界面、球员选择界面、球队选择界面、单个球员信息显示界面、全部球员数据统计界面。界面跳转如图5.2-1所示。

球队信息显示界面

球队排名界面

单个球员信息显示界面

全部球员数据统计界面

主界面

球队选择界面

球员选择界面

图5.2-1 用户界面跳转

用户界面类如图5.2-2所示。

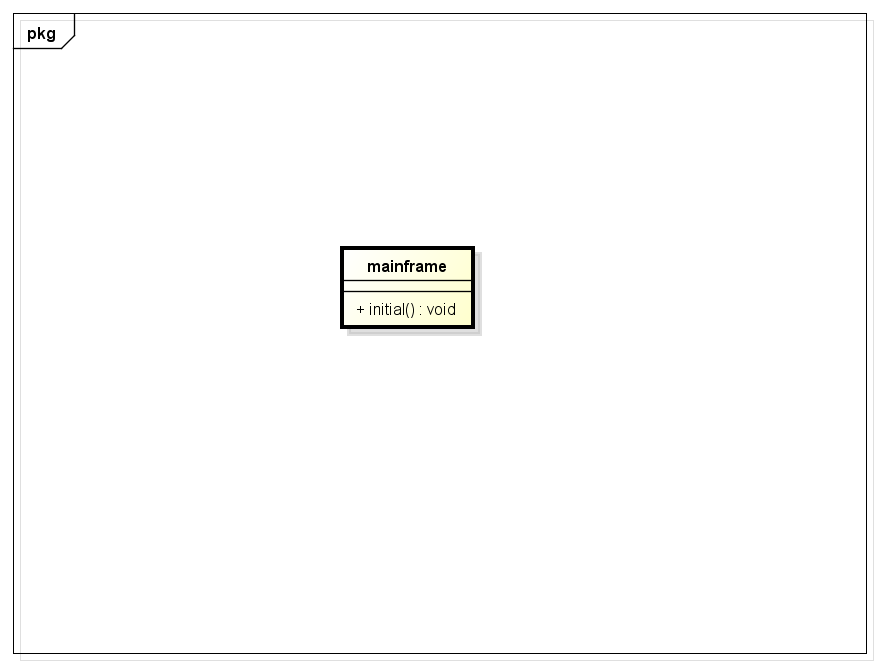


图5.2-2 用户界面类

### **5.2.1**用户界面层模块的职责

如表5.2.1-1所示为用户界面层模块的职责。

表5.2.1-1 用户界面层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| mainframe | 界面frame，负责界面跳转 |

### **5.2.2**用户界面层模块的接口规范

用户界面层模块的接口规范如表5.2.2-1所示。

表5.2.2-1 用户界面层模块的接口规范

|  |  |  |
| --- | --- | --- |
| mainframe | 语法 | init(args[]) |
| 前置条件 | 无 |
| 后置条件 | 显示主界面 |

用户界面层需要的服务接口如表5.2.2-2所示。

|  |  |
| --- | --- |
| 服务名 | 服务 |
| blservice.playerblservice | 球员模块的业务逻辑接口 |
| blservice.teamblservice | 球队模块的业务逻辑接口 |

### **5.2.3**用户界面模块设计原理

用户界面利用Java的Swing和AWT库来实现。

## **5.3**业务逻层的分解

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，player对象负责处理球员的业务逻辑；team对象负责处理球队的业务逻辑。业务逻辑层的设计如图5.3-1所示。

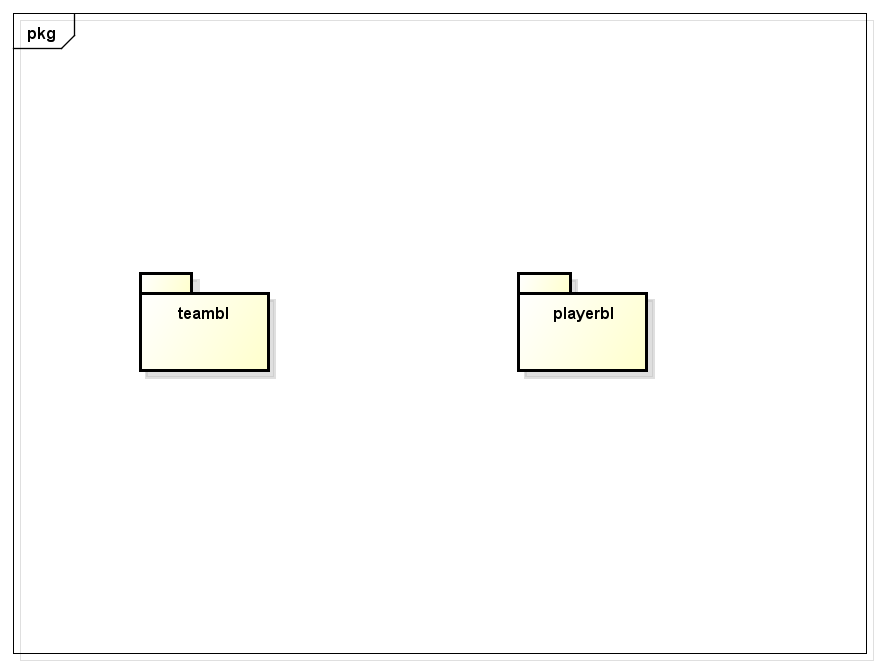


图5.3-1 业务逻辑层

### **5.3.1**业务逻辑层模块的职责

业务逻辑层模块的职责如表5.3.1-1所示。

表5.3.1-1 业务逻辑层模块的职责

|  |  |
| --- | --- |
| 模块 | 职责 |
| playerbl | 负责实现查看球员信息所需要的服务 |
| teambl | 负责实现查看球队信息所需要的服务 |

### **5.3.2**业务逻辑层模块的接口规范

表5.3.2-1 逻辑层playerbl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerRank.**getAllPlayer** | 语法 | public ArrayList<String> getAllPlayer(String teamName) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球员的姓名 |
| PlayerRank.**getPlayerdata** | 语法 | public PlayerVO getPlayerdata(String playerName) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球员的数据 |
| PlayerRank.getPlayerinfo | 语法 | public PlayerinfoVO getPlayerinfo(String playerName) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球员的基本信息 |
| PlayerRank.getAllPlayerdata | 语法 | public ArrayList<PlayerVO> getAllPlayerdata(String key,String order) |
| 前置条件 | 无 |
| 后置条件 | 根据条件获取所有球员的数据 |
| PlayerRank. getFirstFifty | 语法 | public ArrayList<PlayerVO> getFirstFifty(String position,String partition,String key) |
| 前置条件 | 无 |
| 后置条件 | 获取前50名球员数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public PlayerPO getPlayerdata(String playerName)throws RemoteException | 提供某个球员的数据 | |
| public PlayerinfoPO getPlayerinfo(String playerName)throws RemoteException | 提供某个球员的信息 | |
| public ArrayList<PlayerPO> getAllPlayerdata(String key,String order)throws RemoteException | 根据某些条件提供所有球员的数据 | |
| public ArrayList<PlayerPO> getSomePlayerdata(String position,String partition,String key,String order)throws RemoteException | 根据分区和某些条件提供球员的数据（不包括各种效率） | |
| ArrayList<PlayerPO> getByEfficiency(ArrayList<PlayerPO> po,String key,String order)throws RemoteException | 根据分区和某些条件提供球员的数据（包括各种效率） | |

表5.3.2-2 逻辑层teambl模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TeamRank. Ranking | 语法 | public ArrayList<TeamPO> Ranking(String condition,String order) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球队的数据并排序（不含各种效率和排名） |
| TeamRank. gettingTeamData | 语法 | public ArrayList<TeamVO> gettingTeamData(String condition, String key,String order) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球队的数据并排序（含各种效率，不含排名） |
| TeamRank. getTeamData | 语法 | public ArrayList<TeamVO> getTeamData(String condition, String key,String order) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球队的数据并排序（含各种效率，含排名） |
| TeamRank. getTeamInfo | 语法 | public TeaminfoVO getTeamInfo(String teamName) |
| 前置条件 | 无 |
| 后置条件 | 获取球队的信息 |
| TeamRank. getTeamData | 语法 | public ArrayList<PlayerVO> public TeamVO getTeamData(String teamName) |
| 前置条件 | 无 |
| 后置条件 | 获取前50名球员数据 |
| TeamRank. getAllTeamdata |  | public ArrayList<TeamVO> getAllTeamdata(String key,String order) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球队的所有数据 |
| 需要的服务（需接口） | | |
| 服务名 | 服务 | |
| public PlayerPO getPlayerdata(String playerName)throws RemoteException | 提供某个球员的数据 | |
| public PlayerinfoPO getPlayerinfo(String playerName)throws RemoteException | 提供某个球员的信息 | |
| public ArrayList<PlayerPO> getAllPlayerdata(String key,String order)throws RemoteException | 根据某些条件提供所有球员的数据 | |
| public ArrayList<PlayerPO> getSomePlayerdata(String position,String partition,String key,String order)throws RemoteException | 根据分区和某些条件提供球员的数据（不包括各种效率） | |
| ArrayList<PlayerPO> getByEfficiency(ArrayList<PlayerPO> po,String key,String order)throws RemoteException | 根据分区和某些条件提供球员的数据（包括各种效率） | |

## **5.4**数据层的分解

数据层主要给业务逻辑层提供数据防伪服务，包括对于持久化数据的增、删、改、查。由于持久化数据的保存可能存在多种形式：Txt文件、序列化文件、数据库等，所示抽象了数据服务。数据层模块的具体描述如图5.4所示。

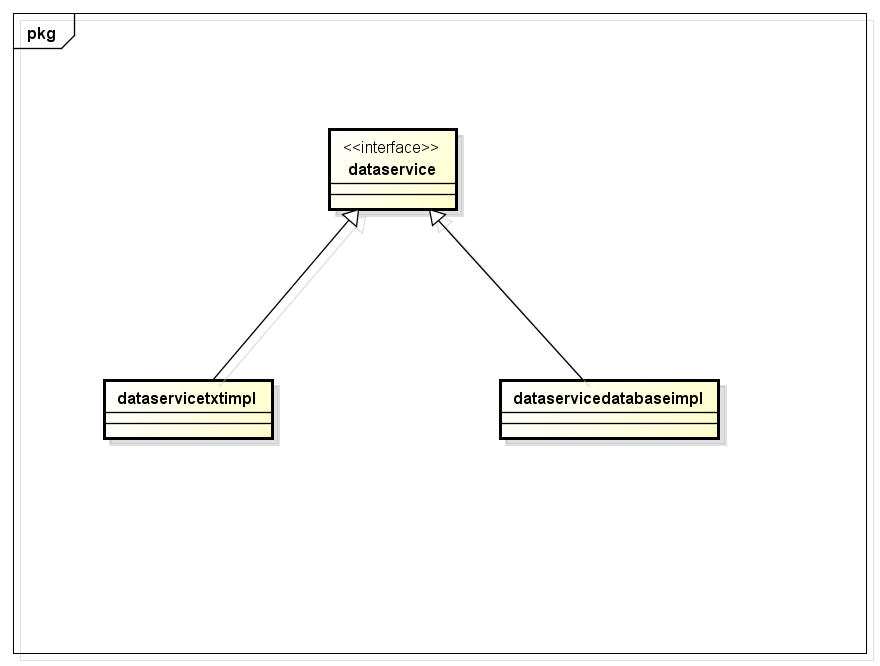


图5.4 数据层模块的描述

### **5.4.****1**数据层模块的职责

数据层模块的职责如表5.4.1所示。

表5.4.1 数据层模块的职责

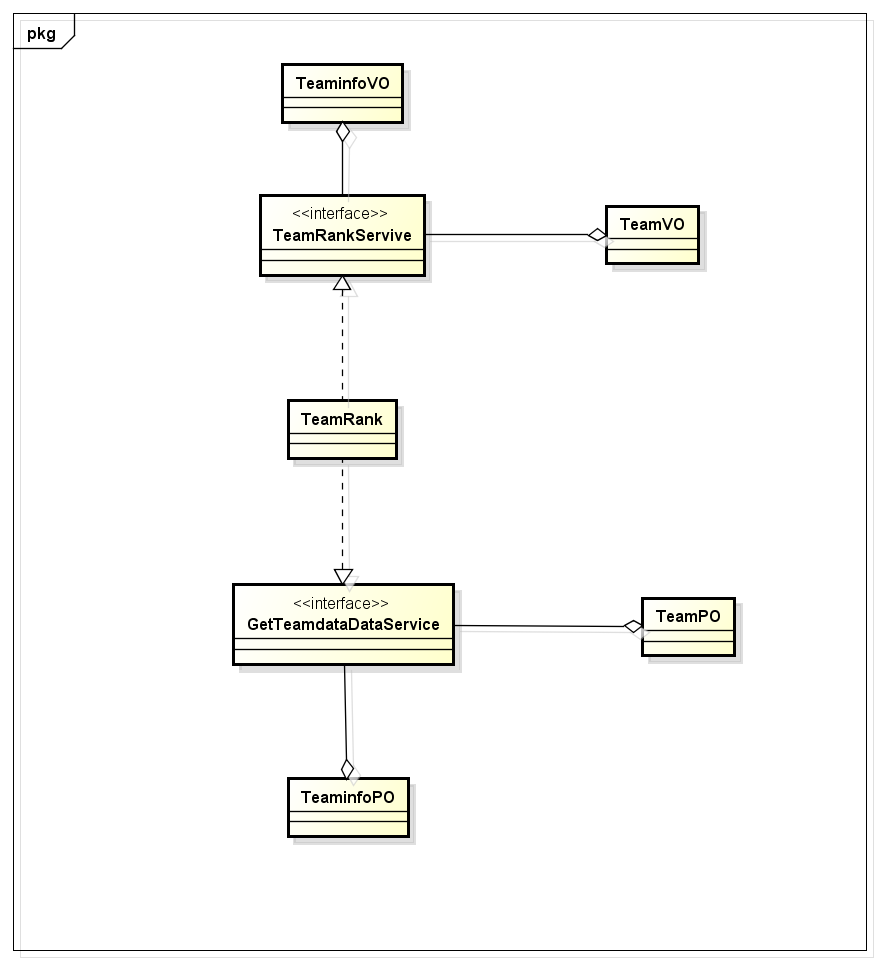
|  |  |
| --- | --- |
| 模块 | 职责 |
| dataservicedatabaseimpl | 持久化数据库的接口，提供集体载入，集体保存、增、删、改、查服务。 |
| dataservicetxtimpl | 基于TXT文件的持久化数据库的接口，提供集体载入、集体保存、增、删、改、查服务 |

### **5.4.2**数据层模块的接口规范

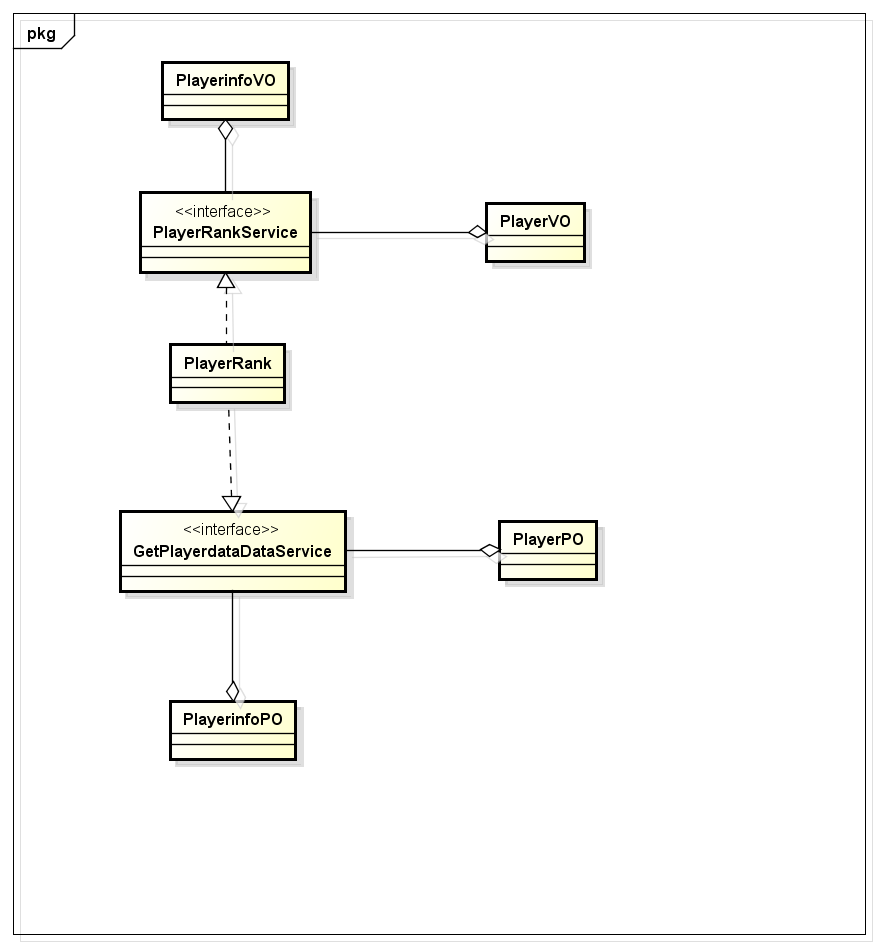
表5.4.2-1 数据层getdata模块的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| getdata.**getPlayerdata** | 语法 | **public** PlayerPO **getPlayerdata**(String **playerName**) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球员的技术统计 |
| getdata.**getPlayerinfo** | 语法 | **public** PlayerinfoPO **getPlayerinfo**(String **playerName**) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球员的基本信息 |
| getdata.**getAllPlayerdata** | 语法 | **public** ArrayList<PlayerPO> **getAllPlayerdata**(String **key**,String **order**) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球员的技术统计 |
| getdata.**getSomePlayerdata** | 语法 | **public** ArrayList<PlayerPO> **getSomePlayerdata**(String **position**,String **partition**,String **key**,String **order**) |
| 前置条件 | 无 |
| 后置条件 | 根据条件获取某些球员的技术统计 |
| getdata.**getTeaminfo** | 语法 | **public** TeaminfoPO **getTeaminfo**(String **teamName**) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球队的基本信息 |
| getdata. **getTeamPlayer** | 语法 | **public** ArrayList<String> **getTeamPlayer**(String **teamName**) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球队所有球员 |
| getdata.**getTeamdata** | 语法 | **public** TeamPO **getTeamdata**(String **teamName**) |
| 前置条件 | 无 |
| 后置条件 | 获取某个球队的技术统计 |
| getdata.**getAllTeamdata** | 语法 | **public** ArrayList<TeamPO> **getAllTeamdata**(String **key**,String **order**) |
| 前置条件 | 无 |
| 后置条件 | 获取所有球队的技术统计 |
| getdata.**getSomeTeamdata** | 语法 | **public** ArrayList<TeamPO> **getSomeTeamdata**(String **condition**,String **key**,String **order**) |
| 前置条件 | 无 |
| 后置条件 | 根据条件获取某些球队的技术统计 |

## **5.5**模块的关键类图



Teambl的关键类图



Playerbl的关键类图

**6．信****息视角**

## **6.1**数据持久化对象

系统的PO类就是对应的相关的实体类，如表6.1所示：

表6.1 PO类介绍

|  |  |
| --- | --- |
| 类名 | 包含的属性 |
| playerinfopo | 球员姓名、队服号码、位置、身高、体重、生日、年龄、球龄、学校 |
| playerpo | 球员姓名、所属球队、参赛场数、先发场数、篮板数、助攻数、在场时间、投篮命中数、投篮出手数、三分命中数、三分出手数、罚球命中数、罚球出手数、进攻数、防守数、抢断数、盖帽数、失误数、犯规数、比赛得分、球队总出手次数、球队总篮板、球队投篮命中数、球队罚球命中数、球队进攻篮板数、球队防守篮板数、所有球员上场时间、球队罚球数、球队失误数、对手总篮板、对手总进攻篮板、对手总防守篮板、对手投篮出手数、对手三分出手数、三分命中率、罚球命中率、效率、GmSc效率、近五场提升率、真实命中率、投篮效率、篮板率、进攻篮板率、防守篮板率、助攻率、抢断率、盖帽率、失误率、使用率、五场前平均得分、近五场平均得分 |
| teaminfopo | 球队名、缩写、所在城市、所在联盟、所在分区、球场、创建时间 |
| teampo | 对手投篮命中数、对手投篮出手数、对手失误数、对手罚球数、对手得分、球队名称、比赛场数、胜利场数、投篮命中数、投篮出手次数、三分命中数、三分出手数、罚球命中数、罚球出手数、进攻篮板数、防守篮板数、对手进攻篮板数、对手防守篮板数、篮板数、助攻数、抢断数、盖帽数、失误数、比赛得分、投篮命中率、罚球命中率、三分命中率、胜率、进攻回合、进攻效率、防守效率、进攻篮板效率、防守篮板效率、抢断效率、助攻效率 |

持久化用户对象如teaminfopo的定义如图6.1。

图6.1 持久化用户对象UserPO的定义

**6.2 database数据表格式**

表6.2-1 matches数据表格式示意表

|  |  |  |
| --- | --- | --- |
| matches | 格式 | date;host/guest;name;opponent;win/lose;total;first;second;third;fourth |
| 举例 | 13-10-29;h;CHI;MIA;l;95;15;18;25;37 |

表6.2-2 playerdata数据表格式示意表

|  |  |  |
| --- | --- | --- |
| playerdata | 格式 | date;team;playername;position;minutes;fieldGoal;fieldGoalAttempts;threepointFieldGoal;threepointFieldGoalAttempts;freeThrow;freeThrowAttempts;offensiveRebound;defensiveRebound;backboard;assit;steal;block;turnOver;foul;scoring |
| 举例 | 13-10-29;CHI;Carlos Boozer;F;32.4;13;18;0;0;5;5;3;4;7;2;1;1;2;4;31 |

表6.2-3 playerinfo数据表格式示意表

|  |  |  |
| --- | --- | --- |
| playerinfo | 格式 | name;number;position;height;weight;birth;age;exp;school |
| 举例 | Aaron Brooks;0;G;6-0;161;JAN 14, 1985;29;5;Oregon |

表6.2-4 teaminfo数据表格式示意表

|  |  |  |
| --- | --- | --- |
| teaminfo | 格式 | name;abbr;city;east/west;partition;court;year |
| 举例 | Hawks;ATL;Atlanta;E;Southeast;PhilipsArena;1949 |