**NBA信息查询平台**

**项目设计文档**

**（迭代一适用）**

**（V1.2）**

**南京大学软件学院**

**羽见青柠工作组（15）**

金翠黄涵倩王宁 (79)严顺宽

**2015年3月11日**

**目录**

更新历史……………………………………………………………………………3

[1、引言](#_引言)………………………………………………………………………………3

1.1编制目的……………………………………………………………………3

1.2词汇表………………………………………………………………………3

1.3参考目录……………………………………………………………………3

[2、逻辑视角](#_3、逻辑视角)…………………………………………………………………………3

[3、组合视角](#_4、组合视角)…………………………………………………………………………5

3.1开发包图……………………………………………………………………5

3.2运行时进程…………………………………………………………………7

3.3物理部署……………………………………………………………………7

[4、接口视角](#_5、接口视角)…………………………………………………………………………8

4.1模块的职责…………………………………………………………………8

4.2用户界面层的分解…………………………………………………………10

4.2.1用户界面层模块的职责……………………………………………11

4.2.2用户界面层模块的接口规范………………………………………11

4.2.3用户界面层模块设计原理…………………………………………12

4..3业务逻辑层的分解…………………………………………………………12

4.3.1业务逻辑层模块的职责……………………………………………12

4.3.2业务逻辑层模块的接口规范………………………………………12

4.4数据层的分解………………………………………………………………19

4.4.1数据层模块的职责…………………………………………………21

4.4.2数据层模块的接口规范……………………………………………22

4.5客户端启动模块的分解……………………………………………………27

4.5.1客户端启动模块的职责……………………………………………28

4.5.2客户端启动模块的接口规范………………………………………28

4.5.3客户端启动模块设计原理…………………………………………28

4.6服务器端启动模块的分解………………………………………………29

4.6.1服务器端启动模块的职责………………………………………29

4.6.2服务器端启动模块的接口规范……………………………………29

4.6.3服务器端启动模块设计原理………………………………………29

4.7客户端网络模块的分解……………………………………………………29

4.8服务器端网络模块的分解…………………………………………………30

[5、信息视角](#信息视角)…………………………………………………………………………31

5.1数据持久化对象…………………………………………………………31

5.2序列化文件表……………………………………………………………………33

# 更新历史

|  |  |  |  |
| --- | --- | --- | --- |
| **修改人员** | **日期** | **变更原因** | **版本号** |
| 金翠 | 2015/3/11 | 文档编写 | V1.0 |
| 严顺宽 | 2015/3/12 | 补充数据层设计说明 | V1.1 |
| 金翠 | 2015/3/14 | 对球员数据层进行增加了接口 | V1.2 |

# 1、引言

**.1.1编制目的**

本报告详细完成对NBA信息查询平台的项目设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

**1.2 词汇表**

|  |  |  |
| --- | --- | --- |
| **词汇名称** | **词汇含义** | **备注** |
| bl | 业务逻辑（business logic） | 无 |
|  |  |  |

**1.3 参考资料**

骆斌，丁二玉，刘钦.软件工程与计算（卷二）——软件开发的技术基础.北京：机械工程出版社，2012.12;

骆宾，刘嘉，张瑾玉，黄蕾，软件工程与计算(卷三)——团队与软件开发实践，北京：机械工业出版社，2012.12

NBA信息查询平台软件需求规格说明文档。

# 2、逻辑视角

在NBA信息查询平台中，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）能够很好地展示整个高层抽象。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如图1所示。



**图1 参照体系结构风格的包图表达逻辑视角**

# 3、组合视角

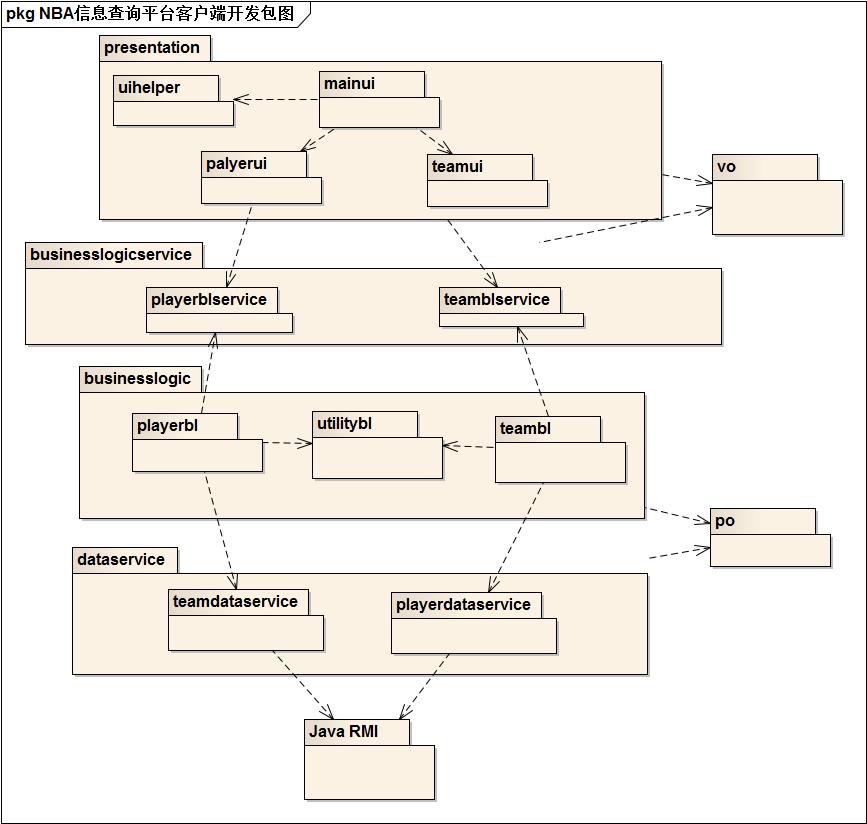
**3.1 开发包图**

NBA信息查询平台的最终开发包设计如表1所示。

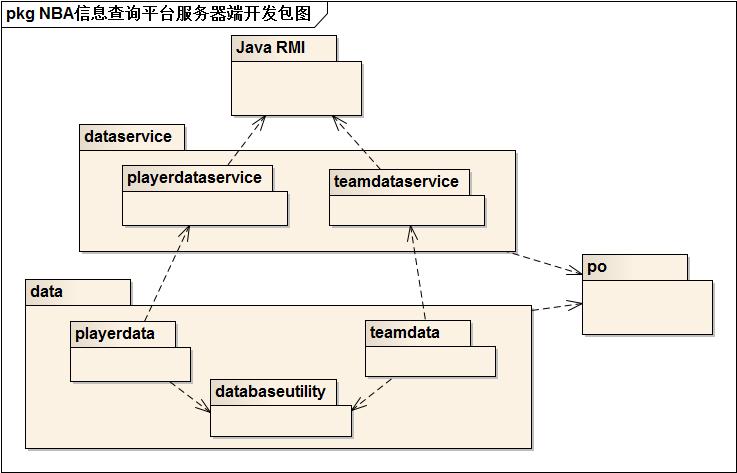
**表1 NBA信息查询平台的最终开发包设计**

|  |  |
| --- | --- |
| **开发（物理）包** | **依赖的其他开发包** |
| **mainui** | **playerui,teamui, vo,uihelper** |
| **playerui** | **playerblservice，界面类库包,vo** |
| **playerblservice** |  |
| **playersbl** | **playerslservice,playerdataservice,po** |
| **playerdataservice** | **java rmi,po** |
| **salesdata** | **databaseutility,po,playerdataservice** |
| **teamui** | **teamblservice,界面类库包** |
| **teamblservice** |  |
| **teambl** | **teamblservice, teamdataservice,po** |
| **teamdataservice** | **java rmi,po** |
| **teamdata** | **java rmi,po,databaseutility** |
| **vo** |  |
| **po** |  |
| **utilitybl** |  |
| **uihelper** |  |
| **界面类库包** |  |
| **java rmi** |  |
| **databaseutility** | **JDBC,mysql** |

NBA信息查询平台客户端开发包图如图2所示，服务器端开发包图如图3所示。



**图3客户器端开发包图**



**图4服务器端开发包图**

**3.2 运行时进程**

在NBA信息查询平台中，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 进程图**

**3.3 物理部署**

NBA信息查询平台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Java RMI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5所示。



**图5 部署图**

# 4、接口视角

**4.1 模块的职责**

客户端模块和服务器模块视图分别如图6和图7所示。客户端各层和服务器端各层的职责分别如表2和表3所示。



**图6 客户端模块视图**



**图7 服务器端模块视图**

**表2 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信息查询平台客户端用户界面 |
| 业务逻辑层 | 对于用户界面的输入进行响应并进行业务处理逻辑 |
| 客户端网络模块 | 利用Java RMI机制查找RMI服务 |

**表3 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 |
| 数据层 | 负责数据库的维护，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 |

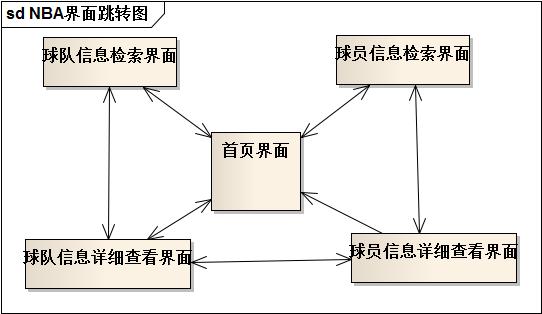
每一层只是使用下方直接接触的层。层与层之间仅仅是通过接口的调用来完成的。层之间调用的接口如表4所示。

**表4 层之间调用的接口**

|  |  |  |
| --- | --- | --- |
| **接口** | **服务调用方** | **服务提供方** |
| TeamBLService  PlayerBLService | 客户端展示层 | 客户端业务逻辑层 |
| TeamDataService  PlayerDataService | 客户端业务逻辑层 | 服务器端数据层 |

**4.2 用户界面层的分解**

根据迭代一需求，系统存在5个用户界面：首页界面、球员个人信息详细查看界面、球队信息详细查看界面、球员信息搜索界面(卡片模式,列表模式，对比模式)、球队信息搜索界面(卡片模式，列表模式，对比模式)。界面跳转如图8所示。



**图8 用户界面跳转**

服务器端和客户端的用户界面设计接口是一致的，只是具体的页面不一样。用户界面类如图9所示。



**图9 用户界面类**

**4.2.1用户界面模块的职责**

如表5所示为用户界面层模块的职责

**表5 用户界面层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| Mainui | 主界面框架，负责界面的显示和界面跳转 |
| UIHelper | 界面辅助模块，负责窗体样式控制 |

**4.2.2用户界面模块的接口规范**

用户界面层模块的接口规范如表6所示。

**表6 用户界面层模块的接口规范**

|  |  |  |
| --- | --- | --- |
| Mainui | 语法 | main (args:String[]) |
| 前置条件 | 无 |
| 后置条件 | 显示startPanel和MainFrame |

用户界面层需要的服务接口如表7所示。

**表7用户界面层模块需要的服务接口**

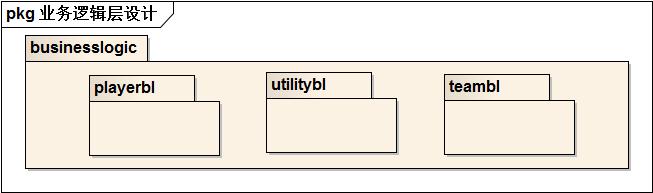
|  |  |
| --- | --- |
| **服务名** | **服务** |
| businesslogicservice.PlayerBLService | 球员列表信息初始化的业务接口 |
| Businesslogicservice.TeamBLService | 球队列表信息初始化的业务接口 |
| businesslogicservice.LogBLService | 界面浏览历史信息初始化的接口 |

**4.2.3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。

**4.3业务逻辑层的分解**

业务逻辑层包括多个针对界面的业务逻辑处理对象。例如，playerbl模块负责处理球员信息检索查看的业务逻辑，teambl负责处理球队信息查看有关的界面的业务逻辑。业务逻辑层的设计如图11所示。

****

**图10 业务逻辑层的设计**

**4.3.1业务逻辑层模块的职责**

业务逻辑层模块的职责如表8所示

**表8 业务逻辑层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| **playerbl** | 负责实现对应于球员信息查看，搜索，筛选所需要的服务 |
| **teambl** | 负责实现所有关于球队信息查看，搜索界面所需要的服务 |
| **utilitybl** | 负责实现业务逻辑层服务器连接，日志存储等共同需要的服务 |

**4.3.2业务逻辑层模块的接口规范**

**表9 playerbl模块的接口规**

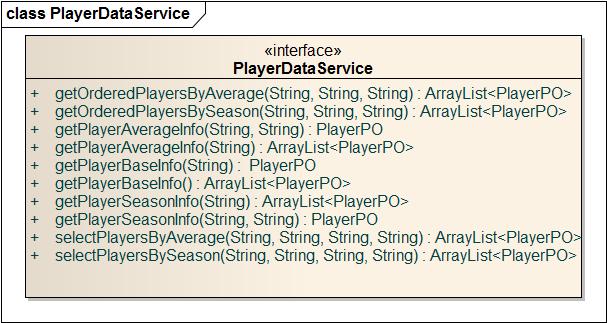
|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Player.getPlayerBaseInfo | 语法 | public ArrayList<PlayerVO> getPlayerBaseInfo() | |
| 前置条件 | 用户请求查看所有球员基本信息 | |
| 后置条件 | 返回所有球员基本信息列表 | |
| Player. getPlayerSeasonInfo | 语法 | public ArrayList<PlayerVO> getPlayerSeasonInfo(String season) | |
| 前置条件 | 用户请求查看所有球员某个赛季的技术数据 | |
| 后置条件 | 返回所有球员该赛季技术数据的列表 | |
| Player. getPlayerAverageInfo | 语法 | public ArrayList<PlayerVO> getPlayerAverageInfo(String season) | |
| 前置条件 | 用户请求查看所有球员某个赛季的场均技术数据 | |
| 后置条件 | 返回所有球员该赛季场均技术数据的列表 | |
| Player. getPlayerBaseInfo | 语法 | public PlayerVO getPlayerBaseInfo(String name) | |
| 前置条件 | 用户请求查看某个球员的基本信息 | |
| 后置条件 | 系统根据输入的用户信息和用户编号更新用户信息 | |
| Player. getPlayerSeasonInfo | 语法 | public PlayerVO getPlayerSeasonInfo(String season, String name) | |
| 前置条件 | 用户请求查看某个球员某个赛季的技术数据 | |
| 后置条件 | 根据输入的球员姓名，返回该球员该赛季的技术数据 | |
| Player. getPlayerAverageInfo | 语法 | public PlayerVO getPlayerAverageInfo(String season, String name) | |
| 前置条件 | 用户请求查看某个球员某个赛季的场均技术数据 | |
| 后置条件 | 根据输入的球员姓名，返回该球员该赛季的场均技术数据 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| PlayerDataService.getPlayerBaseInfo() | | | 得到所有球员的基本信息 |
| PlayerDataService.getPlayerSeasonInfo(String season) | | | 得到所有球员的某赛季的技术数据 |
| PlayerDataService.getPlayerBaseInfo(String name) | | | 得到某位球员的基本信息 |
| PlayerDataService.getPlayerSeasonInfo(String season, String name) | | | 得到某位球员某个赛季的技术数据 |
| PlayerDataService.getPlayerAverageInfo(String season, String name) | | | 得到某位球员的某个赛季的场均技术数据 |
| PlayerDataService.getOrderedPlayersBySeason(String season,String condition, String order) | | | 得到根据某项数据对球员的赛季技术数据排序的所有球员技术数据 |
| PlayerDataService.getPlayerAverageInfo(String season) | | | 得到所有球员的某赛季的场均技术数据 |
| PlayerDataService.getOrderedPlayersByAverage(String season,String condition, String order) | | | 得到根据某项数据对球员的某赛季场均技术数据排序的所有球员技术数据 |
| PlayerDataService.selectPlayersBySeason(String season,String column, String word) | | | 得到根据某项数据筛选出的某个赛季的技术排名前50的球员列表 |
| PlayerDataService.selectPlayersByAverage(String season,String column, String word) | | | 得到根据某项数据筛选出的某个赛季的场均技术排名前50的球员列表 |

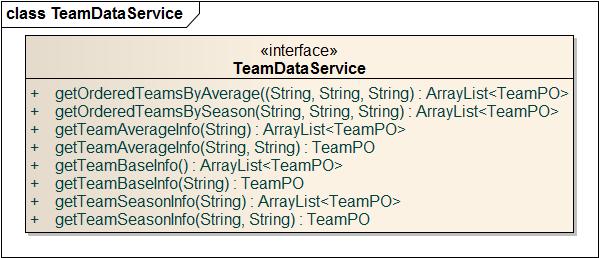
**表10teambl模块的接口规**

|  |  |  |  |
| --- | --- | --- | --- |
| 提供的服务（供接口） | | | |
| Team. getTeamBaseInfo | 语法 | public ArrayList<TeamVO> getTeamBaseInfo() | |
| 前置条件 | 用户请求查看所有球队基本信息 | |
| 后置条件 | 返回所有球队基本信息列表 | |
| Team. getTeamSeasonInfo | 语法 | public ArrayList<TeamVO> getTeamSeasonInfo(String season) | |
| 前置条件 | 用户请求查看所有球队某个赛季的技术数据 | |
| 后置条件 | 返回所有球队该赛季技术数据的列表 | |
| Team. getTeamAverageInfo | 语法 | public ArrayList<TeamVO> getTeamAverageInfo(String season) | |
| 前置条件 | 用户请求查看所有球队某个赛季的场均技术数据 | |
| 后置条件 | 返回所有球队该赛季场均技术数据的列表 | |
| Team. getTeamBaseInfo | 语法 | public TeamVO getTeamBaseInfo(String name) | |
| 前置条件 | 用户请求查看某个球队的基本信息 | |
| 后置条件 | 系统根据输入的用户信息和用户编号更新用户信息 | |
| Team. getTeamSeasonInfo | 语法 | public TeamVO getTeamSeasonInfo(String season, String name) | |
| 前置条件 | 用户请求查看某支球队某个赛季的技术数据 | |
| 后置条件 | 根据输入的球队姓名，返回该球队该赛季的技术数据 | |
| Team. getTeamAverageInfo | 语法 | public TeamVO getTeamAverageInfo(String season, String name) | |
| 前置条件 | 用户请求查看某支球队某个赛季的场均技术数据 | |
| 后置条件 | 根据输入的球队名，返回该球队该赛季的场均技术数据 | |
| 需要的服务（需接口） | | | |
| 服务名 | | | 服务 |
| TeamDataService. getTeamBaseInfo() | | | 得到所有球队的基本信息 |
| TeamDataService. getTeamSeasonInfo(String season) | | | 得到所有球队的某赛季的技术数据 |
| TeamDataService. getTeamBaseInfo(String name) | | | 得到某支球队的基本信息 |
| TeamDataService. getTeamSeasonInfo(String season, String name) | | | 得到某支球队某个赛季的技术数据 |
| TeamDataService. getTeamAverageInfo(String season, String name) | | | 得到某支球队的某个赛季的场均技术数据 |
| TeamDataService.getOrderedTeamsBySeason(String season, String condition, String order) | | | 得到根据某项数据对球队的赛季技术数据排序的所有球队技术数据 |
| TeamDataService. getTeamAverageInfo(String season) | | | 得到所有球队的某赛季的场均技术数据 |
| TeamDataService. getOrderedTeamsByAverage(String season, String condition, String order) | | | 得到根据某项数据对球队的某赛季场均技术数据排序的所有球队技术数据 |

**4.4数据层的分解**

数据层主要给业务逻辑层提供数据访问服务。包括对于持久化数据的保存和读取。这些服务由DataService接口提供。在本系统中我们采用数据库保存持久化数据。具体如图11—12所示。





**图11-12 数据层模块（PlayerDataService和TeamDataService）的描述**

**4.4.1数据层模块的职责**

数据层模块的职责如表11所示。

**表11数据层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| PlayerDataService | 持久化外部文件的接口，提供集体载入、集体保存、根据要求查找等服务 |
| TeamDataService | 持久化外部文件的接口，提供集体载入、集体保存、根据要求查找等服务 |

**4.4.2数据层模块的接口规范**

数据库模块的接口规范如表12-13所示

**表12 数据层模块（PlayerDataService）的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| PlayerDataService.getPlayerBaseInfo | 语法 | public ArrayList<PlayerPO> getPlayerBaseInfo() |
| 前置条件 | 数据库中存在球员信息记录 |
| 后置条件 | 返回所有球员的基本信息 |
| PlayerDataService.getPlayerSeasonInfo | 语法 | public ArrayList<PlayerPO> getPlayerSeasonInfo(String season) |
| 前置条件 | 数据库中存在球员比赛数据记录 |
| 后置条件 | 返回所有球员某个赛季的技术数据 |
| PlayerDataService.getPlayerBaseInfo | 语法 | public PlayerPO getPlayerBaseInfo(String name) |
| 前置条件 | 该球员信息存在 |
| 后置条件 | 返回该球员的基本信息 |
| PlayerDataService.getPlayerSeasonInfo | 语法 | public PlayerPO getPlayerSeasonInfo(String season, String name) |
| 前置条件 | 存在该球员的比赛技术数据记录 |
| 后置条件 | 返回该球员该赛季的技术数据 |
| PlayerDataService.getPlayerAverageInfo | 语法 | public PlayerPO getPlayerAverageInfo(String season, String name) |
| 前置条件 | 存在该球员的比赛技术数据 |
| 后置条件 | 返回该球员该赛季的场均技术数据 |
| PlayerDataService.getOrderedPlayersBySeason | 语法 | public ArrayList<PlayerPO> getOrderedPlayersBySeason(String season,String condition, String order) |
| 前置条件 | 存在所有球员的该赛季的技术数据 |
| 后置条件 | 返回排好序的该赛季的所有球员技术数据 |
| PlayerDataService.getPlayerAverageInfo | 语法 | public ArrayList<PlayerPO> getPlayerAverageInfo(String season) |
| 前置条件 | 存在所有球员的该赛季的场均技术数据 |
| 后置条件 | 返回所有球员的该赛季的场均技术数据 |
| PlayerDataService.getOrderedPlayersByAverage | 语法 | public ArrayList<PlayerPO> getOrderedPlayersByAverage(String season,String condition, String order) |
| 前置条件 | 存在所有球员的该赛季的场均技术数据 |
| 后置条件 | 返回排好序该赛季的所有球员技术数据 |
| PlayerDataService.selectPlayersBySeason | 语法 | public ArrayList<PlayerPO> selectPlayersBySeason(String season,String column, String word) |
| 前置条件 | 存在所有球员的该赛季的技术数据 |
| 后置条件 | 返回某项数据前50名的球员信息 |
| PlayerDataService.selectPlayersByAverage | 语法 | public ArrayList<PlayerPO> selectPlayersByAverage(String season,String column, String word) |
| 前置条件 | 存在所有球员的该赛季的场均技术数据 |
| 后置条件 | 返回某项数据前50名的球员信息 |

**表13 数据层模块（TeamDataService）的接口规范**

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| TeamDataService. getTeamBaseInfo | 语法 | public ArrayList<TeamPO> getTeamBaseInfo() |
| 前置条件 | 数据库中存在球队信息记录 |
| 后置条件 | 返回所有球队的基本信息 |
| TeamDataService. getTeamSeasonInfo | 语法 | public ArrayList<TeamPO> getTeamSeasonInfo(String season) |
| 前置条件 | 数据库中存在球队比赛数据记录 |
| 后置条件 | 返回所有球队某个赛季的技术数据 |
| TeamDataService. getTeamBaseInfo | 语法 | public TeamPO getTeamBaseInfo(String name) |
| 前置条件 | 该球队信息存在 |
| 后置条件 | 返回该球队的基本信息 |
| TeamDataService. getTeamSeasonInfo | 语法 | public TeamPO getTeamSeasonInfo(String season, String name) |
| 前置条件 | 存在该球队的比赛技术数据记录 |
| 后置条件 | 返回该球队该赛季的技术数据 |
| TeamDataService. getTeamAverageInfo | 语法 | public TeamPO getTeamAverageInfo(String season, String name) |
| 前置条件 | 存在该球队的比赛技术数据 |
| 后置条件 | 返回该球队该赛季的场均技术数据 |
| TeamDataService. getOrderedTeamsBySeason | 语法 | public ArrayList<TeamPO> getOrderedTeamsBySeason(String season, String condition, String order) |
| 前置条件 | 存在所有球队的该赛季的技术数据 |
| 后置条件 | 返回排好序的该赛季的所有球队技术数据 |
| TeamDataService. getTeamAverageInfo | 语法 | public ArrayList<TeamPO> getTeamAverageInfo(String season) |
| 前置条件 | 存在所有球队的该赛季的场均技术数据 |
| 后置条件 | 返回所有球队的该赛季的场均技术数据 |
| TeamDataService. getOrderedTeamsByAverage | 语法 | public ArrayList<TeamPO> getOrderedTeamsByAverage(String season, String condition, String order) |
| 前置条件 | 存在所有球队的该赛季的场均技术数据 |
| 后置条件 | 返回排好序该赛季的所有球队技术数据 |

**4.5客户端启动模块的分解**

客户端启动模块逻辑上是一个横跨客户端各层的模块，为客户端各层提供初始化启动服务。客户端启动模块主要包含主程序类ClientStarter。主程序类是整个程序的入口，负责整个程序的展示层初始化、业务逻辑层初始化和网络初始化。其结构如图13所示。



**图13客户端启动模块结构图**

**4.5.1 客户端启动模块的职责**

如表14所示为客户端启动模块的职责。

**表14客户端启动模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| ClientStarter | 程序入口。负责展示层初始化、业务逻辑层初始化和网络初始化。 |

**4.5.2客户端启动模块的接口规范**

客户端启动模块的接口规范如表15所示。

**表15 客户端启动模块的接口规范**

|  |  |  |
| --- | --- | --- |
| ClientStarter | 语法 | main（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 根据配置初始化好相应的网络连接并进行展示层和业务逻辑层初始化。 |

客户端启动模块需要的服务接口如表16所示。

**表16 客户端启动模块的服务接口**

|  |  |
| --- | --- |
| **服务名** | **服务** |
| java.rmi.LocateRegistry.createRegistry() | 启动RMI注册服务 |
| java.rmi.Naming.rebind() | 注册RMI服务 |

**4.5.3客户端启动模块设计原理**

启动模块是一个独立于分层体系之外的模块，可以直接访问各层的接口，完成系统启动的需求。

**4.6服务器端启动模块的分解**

服务器端启动模块逻辑上是一个横跨服务器端各层的模块，为服务器端各层提供初始化启动服务。服务器端启动模块主要包含主程序类SeverStarter。主程序类是整个程序的入口，负责整个程序的网络初始化和外部文件初始化。其结构如图14所示。



**图14 服务器端启动模块结构图**

**4.6.1 服务器端启动模块的职责**

如17所示为服务器端启动模块的职责。

**表17 服务器端启动模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| ServerStarter | 程序入口。负责网络初始化和外部文件初始化。 |

**4.6.2 服务器端启动模块的接口规范**

服务器端启动模块的接口规范如表18所示。

**表18服务器端启动模块的接口规范**

|  |  |  |
| --- | --- | --- |
| ServerStarter | 语法 | main（args:String[]） |
| 前置条件 | 无 |
| 后置条件 | 根据配置初始化好相应的网络连接和进行序列化文件初始化。 |

服务器端启动模块需要的服务接口如表19所示。

**表19服务器端启动模块的服务接口**

|  |  |
| --- | --- |
| **服务名** | **服务** |
| java.rmi.LocateRegistry.createRegistry() | 启动RMI注册服务 |
| java.rmi.Naming.rebind() | 注册RMI服务 |

**4.6.3服务器端启动模块设计原理**

启动模块是一个独立于分层体系之外的模块，可以直接访问各层的接口，完成系统启动的需求。

**4.7客户端网络模块的分解**

由于数据是存储在服务器上的，客户端访问数据必须通过网络通信层，具体是由客户端的业务逻辑层和服务器端的数据层通过网络通信层进行通信。如图15所示。

具体网络通信层的实现是通过远程方法调用RMI来实现的。



**图15客户端数据通信层模块的描述**

其实网络通信层的实际运作代码是由RMI API所完成的，在客户端我们只需要部署RMI服务的Stub文件和Java RMI类库即可。如20所示。

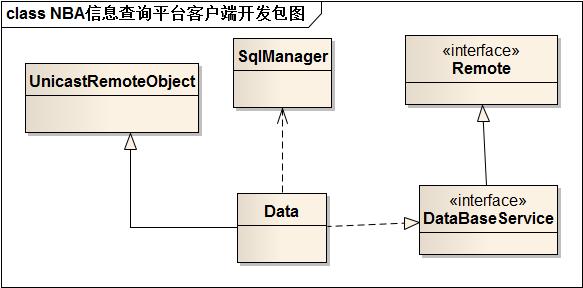
**表20网络通信层模块的职责**

|  |  |
| --- | --- |
| **模块** | **职责** |
| IDataRemoteService\_Stub | 给业务逻辑层提供服务接口 |
| Java RMI 类库 | 实现RMI机制 |

**4.8服务器端网络模块的分解**

由于数据是存储在服务器上的，客户端访问数据必须通过网络通信层，具体是由客户端的业务逻辑层和服务器端的数据层通过网络通信层进行通信。

具体服务器端模块中只需要包含Java RMI类库即可。但是其数据层实现RMI的类需要符合以下条件：

****

**图16 服务器数据层实现RMI的类的描述**

DatabaseService 是被业务逻辑层调用的接口。为了实现RMI， DatabaseService需要继承Remote接口，而数据层中的实现这个接口的类\*\*Data都需要继承UnicastRemoteObject类。

以下是服务器网络模块的RMI详细的规范：

**表21服务器网络模块的RMI接口规范**

|  |  |
| --- | --- |
| **类/接口** | **规范** |
| 远程服务接口 | extends Remote，方法 throws RemoteException |
| 远程服务的实现类 | extendsUnicastRemoteObject |

**表22服务器网络模块的RMI接口返回值的规范**

|  |  |
| --- | --- |
| **类/接口** | **规范** |
| 作为远程引用传递 | 该类需要extends Remote，方法 throws RemoteException |

**5、信息视角**

**5.1 数据持久化对象**

系统的PO类就是对应的相应的实体类，在此只做简单的介绍。

* PlayerPO类包含球员的基本信息和技术数据。

持久化用户对象UserPO的定义如下所示。

public class PlayerPO implements Serializable {

private static final long serialVersionUID = 1L;

private int id;// 编号

private String name;// 球员名称

private int number;// 球衣号码

private String position;// 位置

private String height;// 身高（英尺-英存）

private int weight;// 体重（英镑）

private String birth;// （月日，年）

private int age;// 年龄

private int exp;// 球龄

private String school;// 毕业学校

private String teamName;// 所属球队

private int playedGames;// 参赛场数

private int gameStartingNum;// 先发场数

private int reboundNum;// 篮板数

private int assistNum;// 助攻数

private String presentTime;// 在场时间

private double shootHitRate;// 投篮命中率

private double threeHitRate;// 三分命中率

private double freeThrowHitRate;// 罚球命中率

private int offenNum;// 进攻数

private int defenNum;// 防守数

private int stealNum;// 抢断数

private int blockNum;// 盖帽数

private int turnOverNum;// 失误数

private int foulNum;// 犯规数

private int score;// 得分

private double efficiency;// 效率

private double recentFiveMatchesUpRate;// 近五场的提升率

private double GmScEfficiencyValue;// GmSc效率值

private double trueHitRate;// 真实命中率

private double shootHitEfficiency;// 投篮效率

private double reboundRate;// 篮板率

private double offenReboundRate;// 进攻篮板率

private double defenReboundRate;// 防守篮板率

private double assistRate;// 助攻率

private double stealRate;// 抢断率

private double blockRate;// 盖帽率

private double turnOverRate;// 失误率

private double usageRate;// 使用率

public PlayerPO(int id, String name, int number, String position,

String height, int weight, String birth, int age, int exp,

String school) {

this.id = id;

this.name = name;

this.number = number;

this.position = position;

this.height = height;

this.weight = weight;

this.birth = birth;

this.age = age;

this.exp = exp;

this.school = school;

}

public int getId() {

return id;

}

public String getName() {

return name;

}

public int getNumber() {

return number;

}

public String getPosition() {

return position;

}

public String getHeight() {

return height;

}

public int getWeight() {

return weight;

}

public String getBirth() {

return birth;

}

public int getAge() {

return age;

}

public int getExp() {

return exp;

}

public String getSchool() {

return school;

}

public String getTeamName() {

return teamName;

}

public int getPlayedGames() {

return playedGames;

}

public int getGameStartingNum() {

return gameStartingNum;

}

public String getPresentTime() {

return presentTime;

}

public int getOffenNum() {

return offenNum;

}

public int getDefenNum() {

return defenNum;

}

public double getEfficiency() {

return efficiency;

}

public double getGmScEfficiencyValue() {

return GmScEfficiencyValue;

}

public double getTrueHitRate() {

return trueHitRate;

}

public double getShootHitEfficiency() {

return shootHitEfficiency;

}

public double getReboundRate() {

return reboundRate;

}

public double getOffenReboundRate() {

return offenReboundRate;

}

public double getDefenReboundRate() {

return defenReboundRate;

}

public double getAssistRate() {

return assistRate;

}

public double getStealRate() {

return stealRate;

}

public double getBlockRate() {

return blockRate;

}

public double getTurnOverRate() {

return turnOverRate;

}

public int getReboundNum() {

return reboundNum;

}

public int getAssistNum() {

return assistNum;

}

public double getShootHitRate() {

return shootHitRate;

}

public double getThreeHitRate() {

return threeHitRate;

}

public double getFreeThrowHitRate() {

return freeThrowHitRate;

}

public int getStealNum() {

return stealNum;

}

public int getBlockNum() {

return blockNum;

}

public int getTurnOverNum() {

return turnOverNum;

}

public int getFoulNum() {

return foulNum;

}

public int getScore() {

return score;

}

public double getRecentFiveMatchesUpRate() {

return recentFiveMatchesUpRate;

}

public double getUsageRate() {

return usageRate;

}

public void setId(int id) {

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public void setNumber(int number) {

this.number = number;

}

public void setPosition(String position) {

this.position = position;

}

public void setHeight(String height) {

this.height = height;

}

public void setWeight(int weight) {

this.weight = weight;

}

public void setBirth(String birth) {

this.birth = birth;

}

public void setAge(int age) {

this.age = age;

}

public void setExp(int exp) {

this.exp = exp;

}

public void setSchool(String school) {

this.school = school;

}

public void setTeamName(String teamName) {

this.teamName = teamName;

}

public void setPlayedGames(int playedGames) {

this.playedGames = playedGames;

}

public void setGameStartingNum(int gameStartingNum) {

this.gameStartingNum = gameStartingNum;

}

public void setPresentTime(String presentTime) {

this.presentTime = presentTime;

}

public void setOffenNum(int offenNum) {

this.offenNum = offenNum;

}

public void setDefenNum(int defenNum) {

this.defenNum = defenNum;

}

public void setEfficiency(double efficiency) {

this.efficiency = efficiency;

}

public void setGmScEfficiencyValue(double gmScEfficiencyValue) {

GmScEfficiencyValue = gmScEfficiencyValue;

}

public void setTrueHitRate(double trueHitRate) {

this.trueHitRate = trueHitRate;

}

public void setShootHitEfficiency(double shootHitEfficiency) {

this.shootHitEfficiency = shootHitEfficiency;

}

public void setReboundRate(double reboundRate) {

this.reboundRate = reboundRate;

}

public void setOffenReboundRate(double offenReboundRate) {

this.offenReboundRate = offenReboundRate;

}

public void setDefenReboundRate(double defenReboundRate) {

this.defenReboundRate = defenReboundRate;

}

public void setAssistRate(double assistRate) {

this.assistRate = assistRate;

}

public void setStealRate(double stealRate) {

this.stealRate = stealRate;

}

public void setBlockRate(double blockRate) {

this.blockRate = blockRate;

}

public void setTurnOverRate(double turnOverRate) {

this.turnOverRate = turnOverRate;

}

public void setReboundNum(int reboundNum) {

this.reboundNum = reboundNum;

}

public void setAssistNum(int assistNum) {

this.assistNum = assistNum;

}

public void setShootHitRate(double shootHitRate) {

this.shootHitRate = shootHitRate;

}

public void setThreeHitRate(double threeHitRate) {

this.threeHitRate = threeHitRate;

}

public void setFreeThrowHitRate(double freeThrowHitRate) {

this.freeThrowHitRate = freeThrowHitRate;

}

public void setStealNum(int stealNum) {

this.stealNum = stealNum;

}

public void setBlockNum(int blockNum) {

this.blockNum = blockNum;

}

public void setTurnOverNum(int turnOverNum) {

this.turnOverNum = turnOverNum;

}

public void setFoulNum(int foulNum) {

this.foulNum = foulNum;

}

public void setScore(int score) {

this.score = score;

}

public void setRecentFiveMatchesUpRate(double recentFiveMatchesUpRate) {

this.recentFiveMatchesUpRate = recentFiveMatchesUpRate;

}

public void setUsageRate(double usageRate) {

this.usageRate = usageRate;

}

}

**图20 持久化用户对象PlayerPO的定义**

**5.2数据库文件列表**

数据库中包含（表定义）detailscores.frm,matches.frm,matchtemp.frm，,playermatchdataaverage.frm,playermatchdataseason.frm,players,frm,records.frm,teammatchdataaverage.frm,teammatchdataseason.frm,teammatchtemp.frm,teams.frm.