宠物系统设计

|  |  |  |  |
| --- | --- | --- | --- |
| 文档状态 | 修改历史 | | |
| [ X ] 草稿  [ ] 讨论版  [ ] 实现版 | 作者 | 日期 | 修改说明 |
| 谢江北 | 2015/6/4 | 创建 |
|  |  |  |
|  |  |  |
| 备注 |  | | |

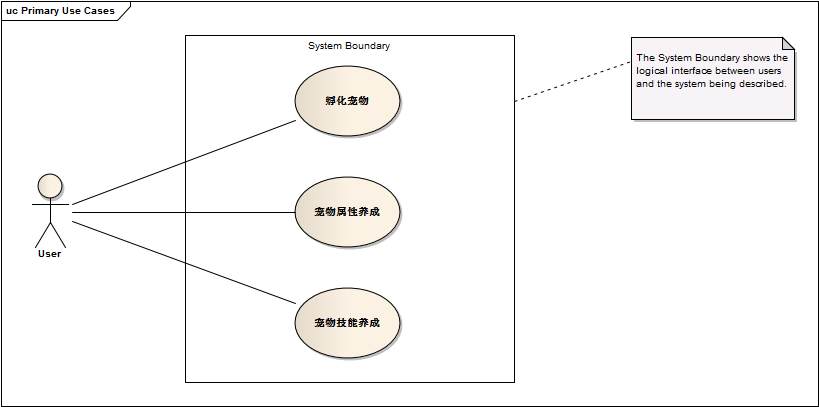
# 需求概述

## 1.1 设计目标

**功能设计目标表：**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **留存率** | **活跃度** | **付费率** | **ARPU** | **核心玩法** | **主要功能** | **辅助功能** | **用户体验** |
|  | YES | YES |  | YES |  |  |  |
| **其它** |  | | | | | | |

## 1.2 系统用例（用例图）

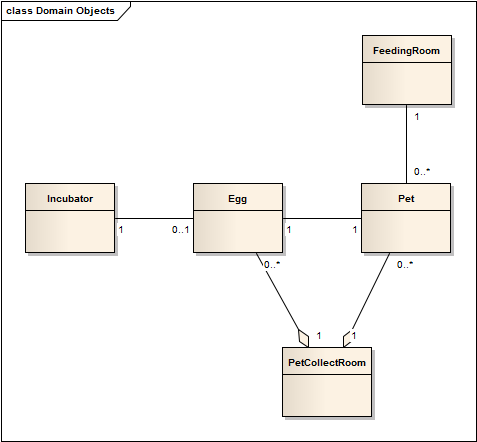


## 1.3 用例说明

宠物系统分为两大的模块：孵化宠物、宠物养成，宠物养成中又包括宠物属性养成和宠物技能养成。

# 领域模型

## 2.1 领域模型类图

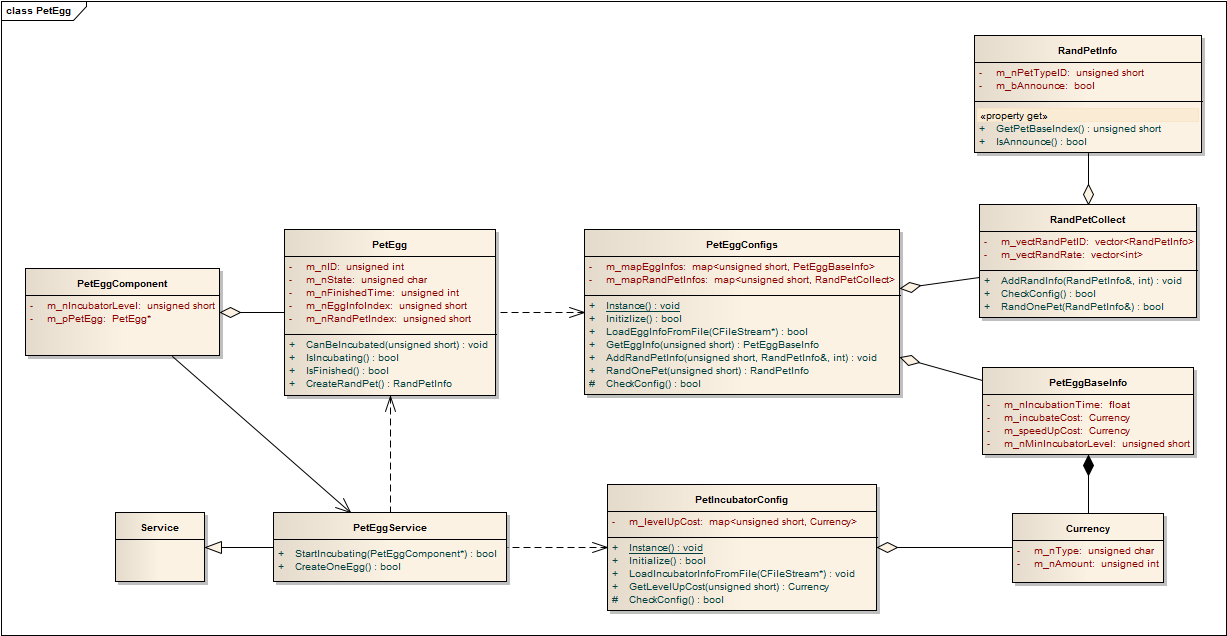


## 2.2 名词解释和对象关系描述

|  |  |  |
| --- | --- | --- |
| 名词 | 说明 | 备注 |
| 宠物收集房间 | 玩家虚拟的一个地方，可以存放宠物蛋和宠物 |  |
| 孵化器 | 通过使用孵化器孵化宠物蛋 |  |
| 宠物蛋 | 宠物蛋通过孵化，得到宠物 |  |
| 宠物 | 宠物可以进行养成 |  |
| 喂养室 | 在喂养室中对宠物进行养成操作 |  |

# 系统设计

## 3.1 宠物蛋类图



## 3.2 宠物蛋类图说明

1.PetEggComponent组件管理孵化器等级、宠物蛋等信息。

2.PetEggService类进行逻辑处理，将逻辑与数据分离。

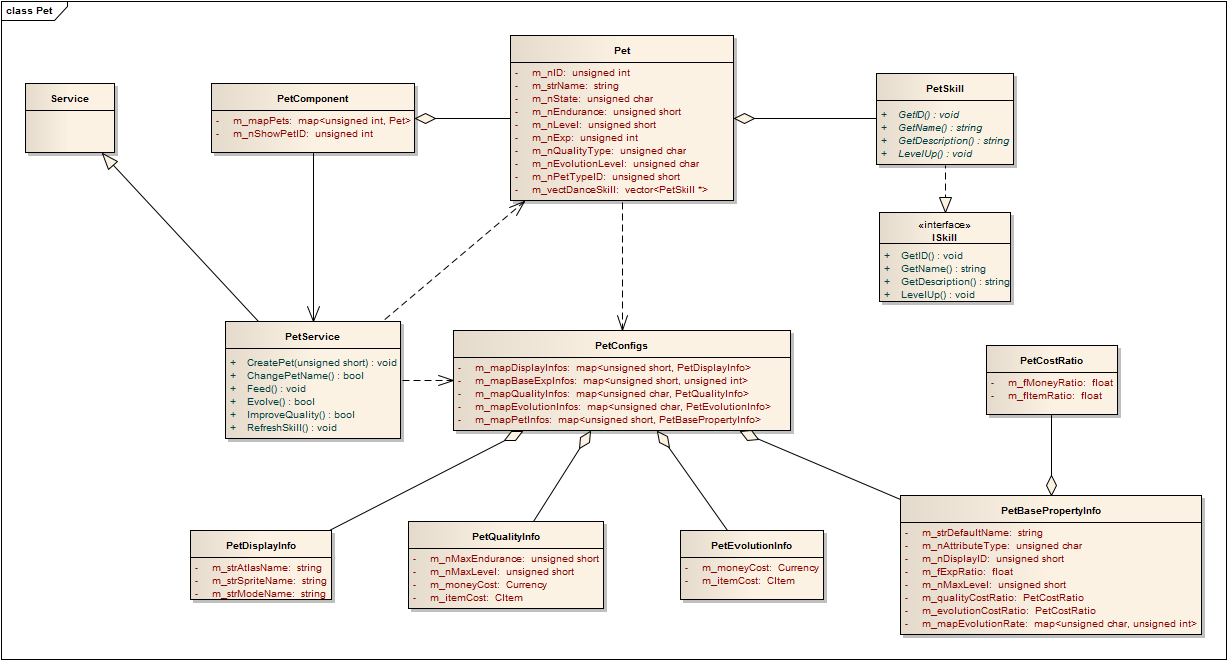
3.PetIncubatorConfig类管理孵化器配置信息。

4.PetEggConfig类管理宠物蛋基础信息和宠物蛋孵化列表。

5.PetEgg类为宠物蛋类，在所有孵化条件满足的情况下生产，在孵化完成并获得宠物后销毁。

6.Currency类、PetEggBaseInfo类、RandPetCollect类和RandPetInfo类为具体配置数据存储类。

## 3.3 宠物类图



## 3.4 宠物类图说明

1.PetComponent组件管理宠物信息及当前显示的宠物。

2.PetService类进行逻辑处理。

3.Pet类为具体的宠物信息。

4.PetConfigs类管理宠物的基础配置信息。

5.PetSkill类为宠物技能基类。

## 3.5 核心功能时序图

对核心业务场景画相应的时序图，并描述关键的业务流程。

## 3.6 接口设计

包括如下几部分，用表格列出：

1. 系统对外的消息接口；
2. 系统内部使用的接口

要求详细列出每个接口的作用、输入输出参数、调用说明等。参考表格如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **接口名称** | Login，账号登陆 | | |
| **输入参数** | | | |
| 参数名称 | 参数类型 | 参数含义 | 备注 |
| username | String | 用户名 |  |
| password | String | 账号密码 | MD5明文密码后得到的16进制32个字符的字符串 |
| zone | integer | 大区编号 | 从1开始，1就是1区 |
| **输出参数** | | | |
| 参数名称 | 参数类型 | 参数含义 | 备注 |
| errno | Integer | 错误码 | 0是成功，非0失败 |
| errmsg | string | 错误描述 |  |

## 3.7 可扩展性设计

对类图做进一步的补充说明，主要是可扩展性的说明。

## 3.8 数据库设计

数据库表如何设计，数据文件如何设计，以表格形式给出。参考如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **表名** | role | 角色表，记录所有角色的基本信息。 | | |
| **字段** | **数据类型** | **可空** | **说明** | **备注** |
| account | int unsigned | NO | 角色所属的账号，对应于player表中的account | 唯一 |
| roleid | int unsigned | NO | 角色id | 主键，自增 |
| rolename | char(50) | NO | 名字 | 唯一 |
| sex | tinyint | NO | 性别 |  |
| birthday | char | NO | 生日，如199096、19901015 |  |
| color | tinyint | NO | 肤色 |  |
| signature | char(100) | NO | 个性签名 |  |
| hot | int unsigned | NO | 人气值 |  |
| pre | int unsigned | NO | 声望属性 |  |
| inte | int unsigned | NO | 智力属性 |  |
| apoint | int unsigned | NO | 尚未分配的属性点数 |  |
| money | int | NO | 拥有的金钱 |  |
| exp | int unsigned | NO | 当前的经验值 |  |
| level | smallint unsigned | NO | 等级 |  |

## 3.9 配置设计

面向策划的配置，如何设计，如何使用。

## 3.10 算法设计

关键算法的流程图，使用软件流程图，不是UML时序图。原因：流程图对于条件判断、循环等有更强的描述能力，能更好地描述复杂的算法。常规的业务流程，可能只需要时序图，不太需要特别的算法设计，这一节可以不写。

## 3.11 性能热点分析

# 风险与异常

## 4.1 异常处理流程

## 4.2 风险点列表