EOS项目设计说明书

# 系统架构设计

## 系统层次图

EOS系统采用层次化结构设计，系统结构图如下所示：



系统架构详细说明如下：

* 数据接口层

直接和文件系统以及数据库通信，为上层模块提供数据库访问和文件存储接口

* 业务逻辑层

业务逻辑的实现层次，对数据进行处理并向用户接口曾提供业务逻辑相关的接口。所有的错误处理也在该层实现。

* 爬虫

爬虫类在业务逻辑曾和数据接口曾之间，该类访问internet得到资源数据，并调用数据层的接口存储资源数据

* 用户接口层

调用业务逻辑层的接口，提供用户以及其他应用程序的访问交互接口。

## 系统包结构

下面给出EOS系统的包结构设计

* 数据接口层
  + DOA包

提供了数据库相关的基本增删改查接口

* + Storage包

提供了和文件系统相关的存储、读取接口。

* 业务逻辑层
  + Business包

包含业务逻辑处理模块，异常处理在该层实现

* 爬虫
  + Crawler包

包含各种资源对应的爬虫类

* 用户接口层
  + Servlet包

提供servlet接口供用户和应用程序使用

# 模块详细设计

## 项目属性配置模块

## 爬虫设计

所有爬虫需要实现一个统一的接口，接口规定如下：

public void Crawl(Project project, List<String> urllist);

参数中project中包含项目基本信息，urllist中包含了需要爬取的资源的url，返回值代表了爬取结果，包括三种状态:

SUCCESS = 1; 表示爬取成功

WAITING = 0; 表示爬取仍然在等待

ERROR = 2; 表示爬取异常并中断

每种资源爬虫的设计细节如下。

### 文档爬虫

### 邮件列表爬虫

### 缺陷库爬虫

### 版本库爬虫

# 数据库表设计

## 数据库表ER图

下面给出系统主要数据库表的ER图形式



## 数据库表结构说明

EOS系统数据库由一个project表和多个资源表组成，每个资源表对应一种类型的资源

### Project表结构说明

Project表存储了每个开源项目的基本信息，具体属性定义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| Uuid | Char | 项目唯一标识 |  |
| Name | Char | 开源项目名称 |  |
| hostUrl | Char | 项目主页地址 |  |
| programminglanguage | Char | 项目编程语言 |  |
| Description | Char | 简短的文本描述 |  |

同时，每个项目还应该包含一个项目—资源表，记录的每个开源项目对应的资源类型和相关描述信息。表属性如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| projectuuid | Char | 开源项目标识 | 外键 |
| type | Char | 资源类型 |  |
| urlListString | Char | 资源爬取的种子地址列表 |  |
| crawler | Char | 爬取该资源的爬虫类 |  |

### 资源表结构说明

每种资源表以该种资源类型命名，表中存储了该种类型资源的所有资源项的原数据信息。每类资源的数据库表至少包含以下表项：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| Project-Uuid | Char | 该项资源对应的项目uuid | 外键 |
| Uuid | Char | 资源uuid |  |
| Filename | Char | 资源文件路径 |  |
| Updatetime | Char | 资源更新时间 |  |

文档资源表包含如下表项

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| projectuuid | Char | 该项资源对应的项目uuid | 外键 |
| uuid | Char | 资源uuid |  |
| subType | Char | 文档的子类型 |  |
| url | Char | 文档的爬取地址 |  |
| name | Char | 文档名称 |  |
| filePath | Char | 资源文件路径 |  |
| Updatetime | Char | 资源更新时间 |  |

邮件列表资源表包含如下表项：

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| projectuuid | Char | 该项资源对应的项目uuid | 外键 |
| uuid | Char | 资源uuid |  |
| Topic | Char | 邮件标题 |  |
| From | Char | 发件人 |  |
| To | Char | 收件人 |  |
| Content | Char | 邮件内容 |  |
| Senddate | Char | 发件日期 |  |

缺陷库资源列表包含一下表项

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **描述** | **备注** |
| projectuuid | Char | 该项资源对应的项目uuid | 外键 |
| uuid | Char | 资源uuid |  |
| type | Char | 缺陷类型 |  |
| summary | Char | 缺陷标题 |  |
| assignee | Char | 负责人 |  |
| reporter | Char | 报告人 |  |
| priority | Char | 优先级 |  |
| status | Int | 缺陷状态 |  |
| resolution |  |  |  |
| created |  |  |  |
| updated |  |  |  |
| description |  | 缺陷描述 |  |
| comments |  | 相关评论 |  |

# 分布式架构设计

## 主控节点

增加：IP/Crawler表

存储方式：配置文件

多对多：Crawler：IP1;IP2;IP3…

例：

TestCrawler=192.168.4.204;192.168.4.205

要增加的功能，分配爬取任务，管理爬虫节点。

分配任务的过程：

CrawlResource（project， type）

根据type从matadata中读取crawler类型

从节点表中找到crawler的ip list

随机分配到节点中的某一个（必须检查检点是否可用，如果节点无响应，返回错误信息，用户来处理）

本身不维护爬虫或者任务信息

需要的时候向爬虫节点询问，有多少任务，任务运行情况如何

每个任务完成或者出错时向主节点发信息

能实时显示任务信息？

功能：维护一个IPlist

分配任务到各个节点

查询节点任务信息

接受任务完成或出错退出信息

网页界面：

项目列表

资源列表

新建项目，填写各项

启动爬虫任务

各个节点状态

## 爬虫节点

基本不变，代码需要重构一下

断点续爬需要支持

如何方便扩展？

增加的功能：

接收爬取任务

完成时提交完成信息

维护爬虫列表

可查询其中爬虫的状态

统一的文件存储接口

先存储文件再存数据库