ACHIEVEIT  
软件架构设计说明书

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | V0.1 |
| 作 者： | G03 |
| 完成日期： | 2020-03-08 |

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V0.1 | G03  曹威杰 | 苏美澄  叶姝晴  曹威杰  赵宁  陶明沺  陈弈君 | 2020.03.02  至2020.03.08 | 初版 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 概述 3](#_Toc34606323)

[2 设计目标和约束 3](#_Toc34606324)

[3 架构设计 3](#_Toc34606325)

[3.1 总体方案 3](#_Toc34606326)

[3.2 架构说明 4](#_Toc34606327)

[3.2.1 架构图及说明 4](#_Toc34606328)

[3.2.2 架构设计关键点 4](#_Toc34606329)

[3.2.3 高可用性设计 5](#_Toc34606330)

[3.2.4 高性能设计 7](#_Toc34606331)

[3.2.5 可扩展性设计 7](#_Toc34606332)

[3.2.6 安全性设计 7](#_Toc34606333)

[3.2.7 其他设计 7](#_Toc34606334)

[4 部署方案 7](#_Toc34606335)

# 概述

本文档将简述AchieveIt项目管理软件的软件设计，主要从系统逻辑架构设计、对象设计和数据库设计三个角度展开，并针对系统的可用性、高性能、拓展性、安全性要求进行讨论设计。

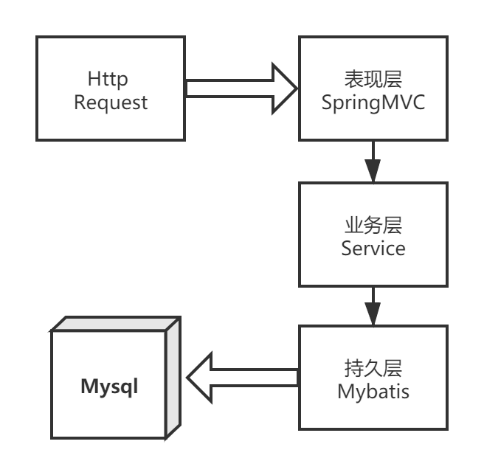
# 设计目标和约束

本软件旨在为公司内部提供一个高效的可用的项目管理系统，以显著提高软件开发的管理效率。因此第一约束为在满足兼容公司原有人事系统、软件缺陷系统、Git/文件/邮件系统和资产系统的前提下提供清晰可用的软件开发管理工作流。其次本软件需要满足一定的安全性和可靠性，保证清晰的权限管理和资产的安全完整。最后考虑到公司的发展，本软件应预留一部分性能空间和设计可拓展性，主要体现在系统的响应时间和自主设计的工作流引擎上。

# 架构设计

## 总体方案

从开发者角度来说，本软件的后端将基于Springboot开发。因此整体框架会采用SSM (SpringMVC + Service + MyBatis)系统架构，使用MSCM(Mapper/Service/Controller/Model)层级设计完成后端的设计开发。系统架构如下图所示：

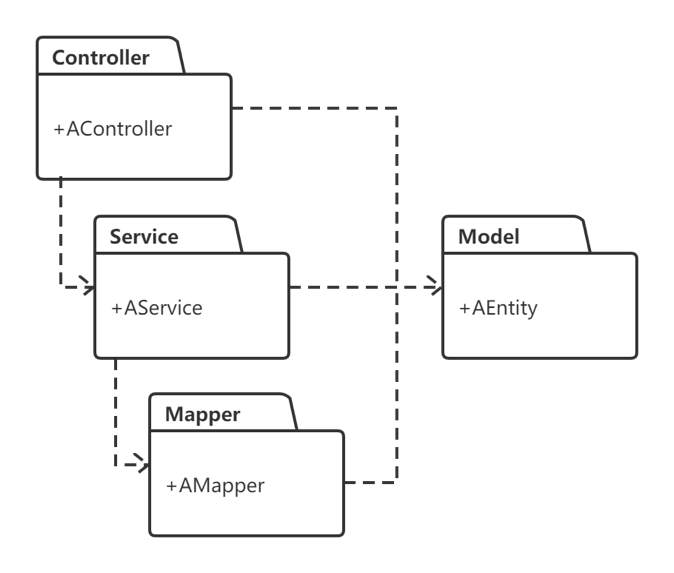


在SSM系统架构中，系统通过Spring将各层进行整合，分别管理持久层、业务层和进行事务控制。而通过Spring管理的表现层将作为软件的接口为前端提供接口，与业务层交互完成整个业务。业务层负责将复杂的业务拆分成多个细粒度的Dao操作，调用Dao获取数据进行封装成为结果。持久层使用Mybatis框架，完成业务层提交的数据库操作。在整个系统框架中，Spring将持有各层依赖的对象的实例，通过依赖注入的方式注入到需要的地方、降低系统分层之间的耦合度。

## 架构说明

### 架构图及说明

在软件架构设计方面，将系统分为MSCM(Mapper/Service/Controller/Model)四个层级进行抽象，抽象层级结构如下面给出的包图所示：



在MSCM设计中，Controller对应SSM框架中的表现层，Service对应框架中的业务层，Mapper对应框架的中的持久层。最后Model层中存放软件的实体类，与数据库中的属性值保持一致。

### 架构设计关键点

MSCM设计的关键点在于职责的划分，按照MSCM的方式分层可以进一步细化软件架构的划分，使各个组件的职责清晰。具体的职责划分如下：

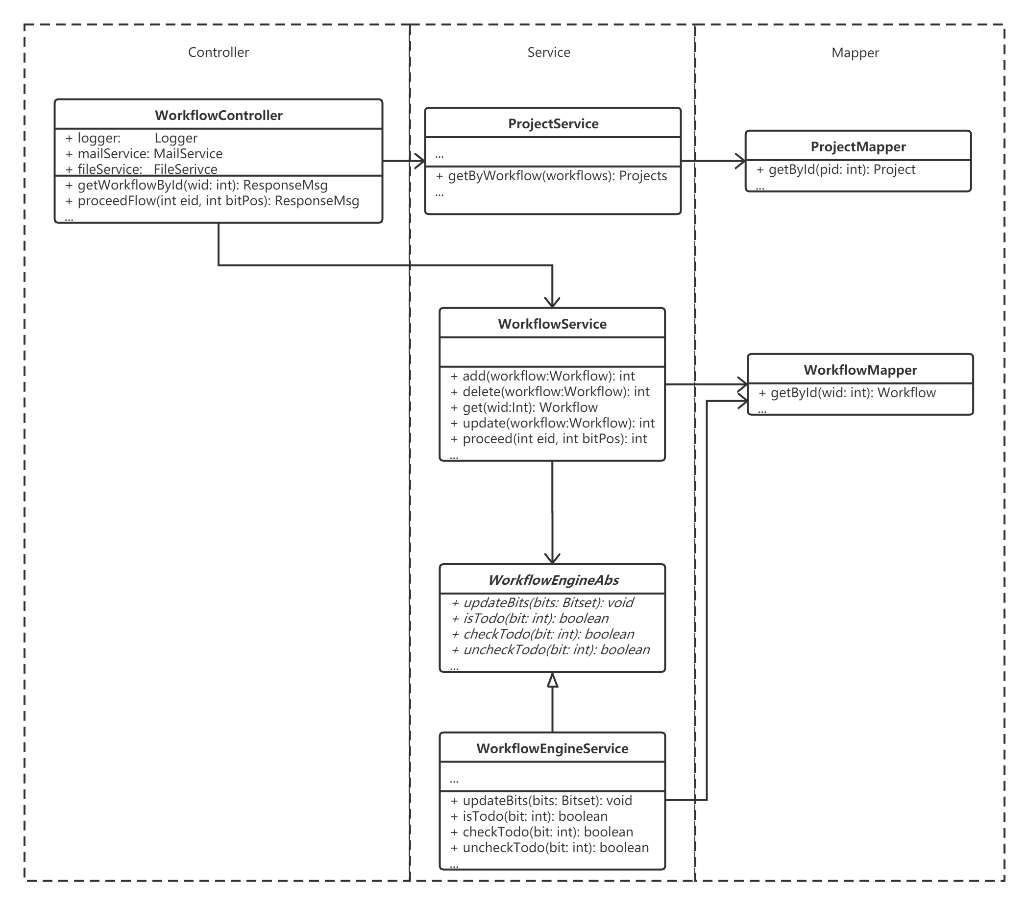
1. Controller层中类主要的职责为：表单校验、控制跳转、调用服务和异常处理。它实质上是软件的API，负责响应控制Http请求调用服务完成业务。
2. Service层中类的主要职责为：校验查询条件、调用持久层、业务逻辑处理、数据封装、异常处理、事务处理。
3. Mapper层中类的主要职责为：执行数据库操作、事务控制、持久化数据、连接数据库。
4. Model层中主要存放与Mysql数据库对应的实体类。

### 高可用性设计

为了满足系统高可用性，需要完成三方面的设计，分别是API抽象、对象设计和数据库设计。首先API抽象如下Mind Map所示：

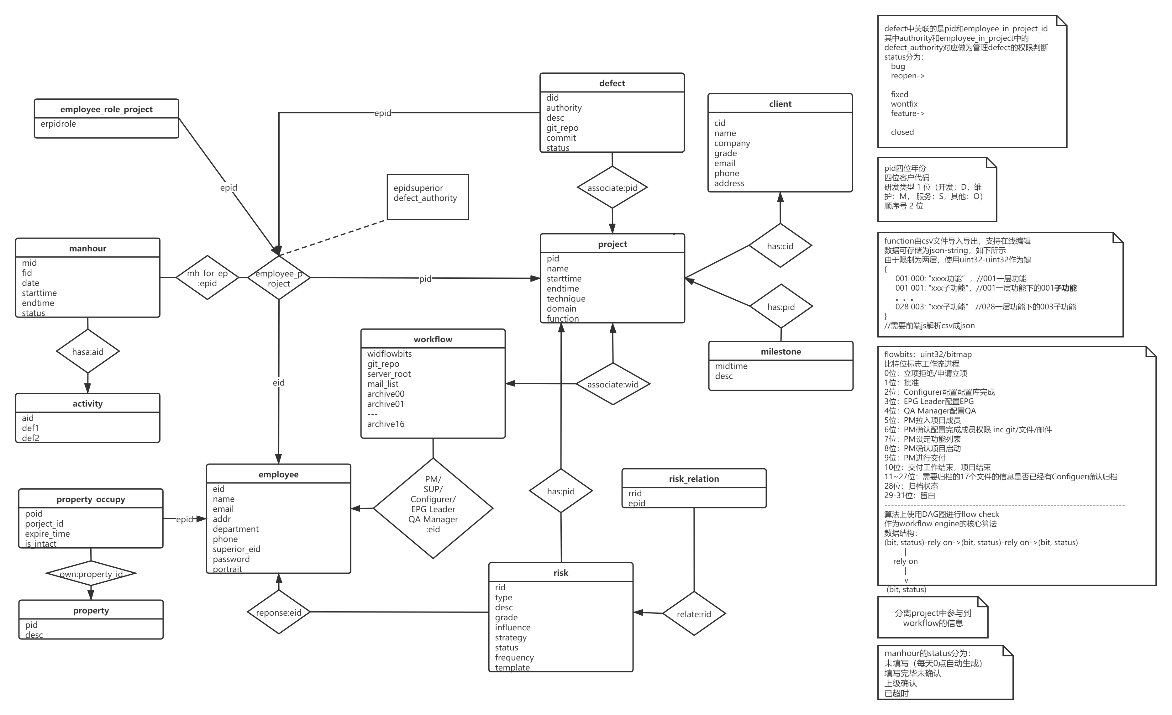


API Mind Map罗列了需求中抽象出来的接口，然后根据责任划分为Controller层分配多个Controller类，依赖对应Service层中提供的逻辑服务，通过Mapper来进行持久层的数据交互完成整个业务。出于篇幅原因，这里只给出了部分对象设计的类图，如下所示：



此图展示了本软件核心业务（工作流业务）的相关类，其中工作流引擎使用了面向对象的Strategy模式来提供核心的工作流服务算法，保证软件需求演变过程中的可拓展性。

为了高可用性，相应的数据库设计也是必须的。数据库使用mysql进行开发，对需求进行抽象和分解保证数据实体具有清晰的依赖关系并不重叠。具体的数据库关系ER图如下所示：



### 高性能设计

初步设计上需要满足在1Core/2GB/1Mb带宽服务器上部署的情况下，达到30人左右规模并发相应时间<5s的需求。通过减小Response的体积和在经济可行范围内提升带宽来完成需求。项目后期考虑分布式部署。

### 可扩展性设计

为了满足多变的需求，如3.2.3的类图所示，将使用面向对象的设计模式来达到可拓展性。现有的拓展被设计为工作流引擎的内核算法拓展。

### 安全性设计

为了让系统达到一定的安全级别，首先将对工作流事务中进行人员权限的校验，此工作置于业务层。其次依赖外挂公司的Git/文件/邮件系统提供公司资产权限的管控。

### **其他设计**

前端使用vue.js框架开发，后端基于Spring boot开发，数据库依赖Mysql提供的服务。

# 部署方案

|  |  |  |
| --- | --- | --- |
| **云服务器组件** | **主要配置** | **具体描述** |
| CPU | 1 Core | / |
| 内存 | 2GB | / |
| 带宽 | 1Mb | 上行加下行总和 |
| Mysql | / | 5.0+ |