<项目名称>

软件架构文档

版本 <1.0>

[注：用方括号括起来并以蓝色斜体（样式=InfoBlue）显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式（样式=Body Text）。]

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File>Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit>Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
| 13/12/2022 | 1.0 | 软件架构文档 初始版本 | 杨冬宁，申艾瑶 |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 技术视图 5

8. 数据视图（可选） 5

9. 核心算法设计（可选） 5

10. 质量属性的设计 5

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

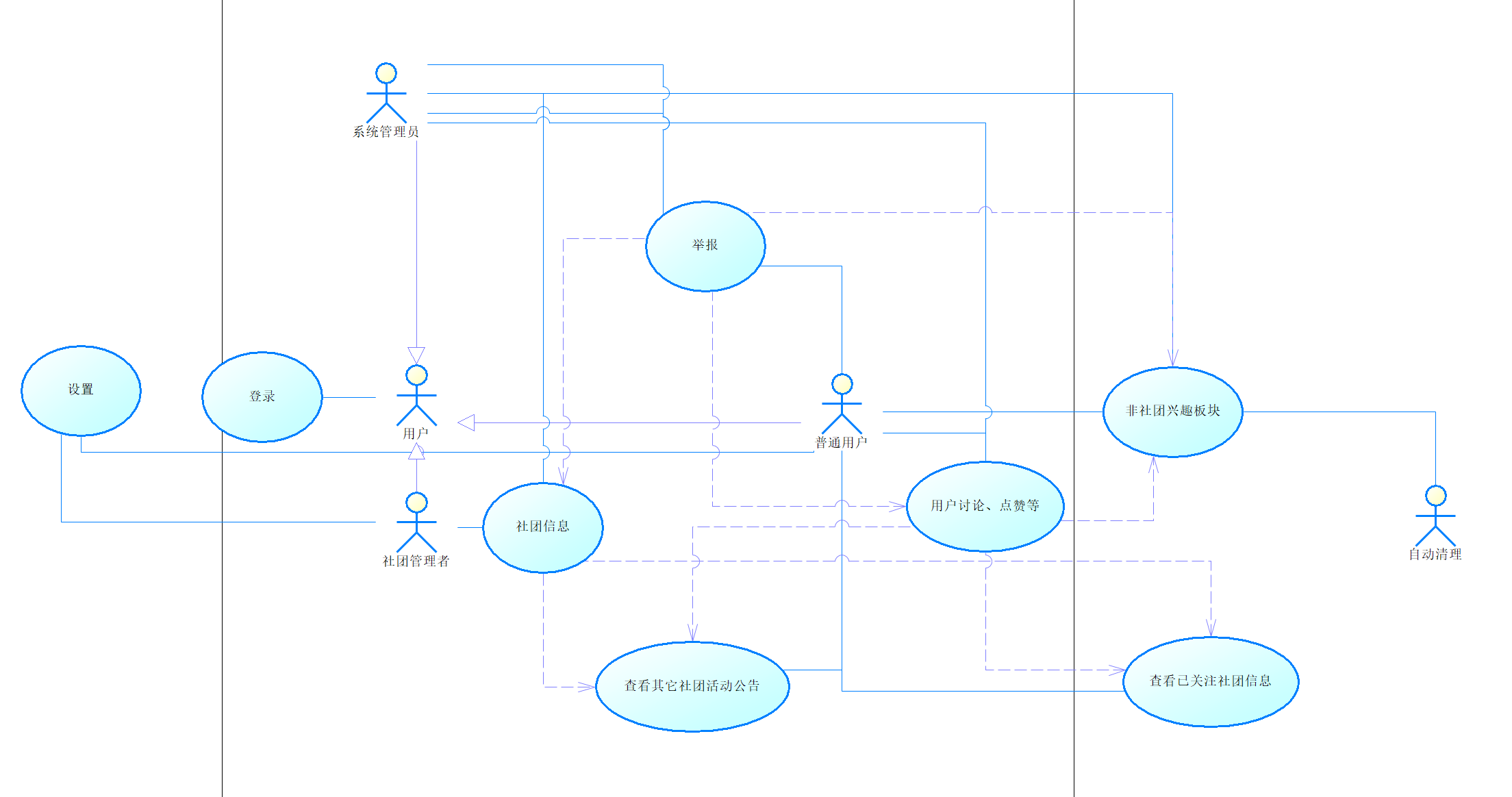
[本节确定此**软件构架文档**在整个项目文档中的作用或目的，并对此文档的结构进行简要说明。应确定此文档的特定读者，并指出他们应该如何使用此文档。]

## 参考资料

[本小节应完整地列出此**软件构架文档**中其他部分所引用的所有文档。每个文档应标有标题、报告号（如果适用）、日期和出版单位。列出可从中获取这些参考资料的来源。这些信息可以通过引用附录或其他文档来提供。]

# 用例视图

[本节列出用例模型中的一些用例或场景，这些用例或场景应体现最终系统中重要的、核心的功能；或是在构架方面涉及范围很广（使用了许多构架元素）；或强调或阐明了构架的某一具体的细微之处。]



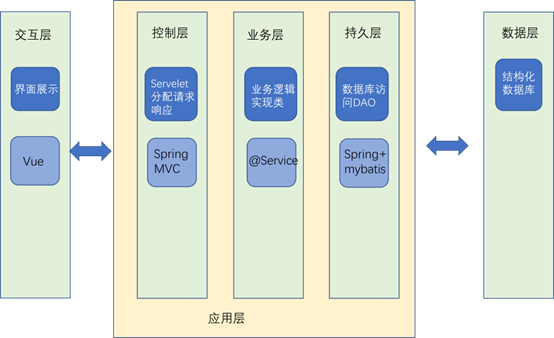
|  |  |
| --- | --- |
| 用例名称 | 讨论 |
| 用例编号 | 1 |
| 执行者 | 普通用户 |
| 用例简述 | 描述用户进行讨论的行为 |
| 触发器 | 当用户提交讨论内容，用例触发 |
| 前置条件 | 用户完成登录并进入相应板块 |
| 基本事件流 | 1.用户将自己的讨论内容提交至系统  2.系统验证用户及讨论内容合法性并作出响应  3.系统上传讨论内容  4.系统生成讨论成功的通知并发送给用户 |
| 扩展事件流 | A-2 若讨论内容非法，系统通知用户并提示重新发布评论 |

# 逻辑视图

[本节说明设计模型在构架方面具有重要意义的部分，例如设计模型被分解为多个子系统和包。而每个重要的包又被分解为多个类和类实用程序。您应该介绍那些在构架方面具有重要意义的类，并说明它们的职责，以及几项非常重要的关系、操作和属性。]

## 概述

[本节说明所采用的架构风格以及整个软件的模块分解和协同情况，画出逻辑架构图，并进行文字描述。]



逻辑架构参考了3 tiers 架构风格和MVC架构风格。

其中交互层是与用户交互的部分，将用户请求传递至控制层，同时将后端的信息渲染后，展示到用户界面。

控制层是分配处理前端请求，并且调用相应的业务层类处理的部分。

业务层是实现主要业务逻辑的部分，处理不同的业务流程，提供用例视图中的各项功能。

持久层是和数据库交互的部分，通过实现不同的DAO类，对数据库中数据进行增删改查的操作。

数据层是结构化数据库，负责数据的存储，提取和变更。

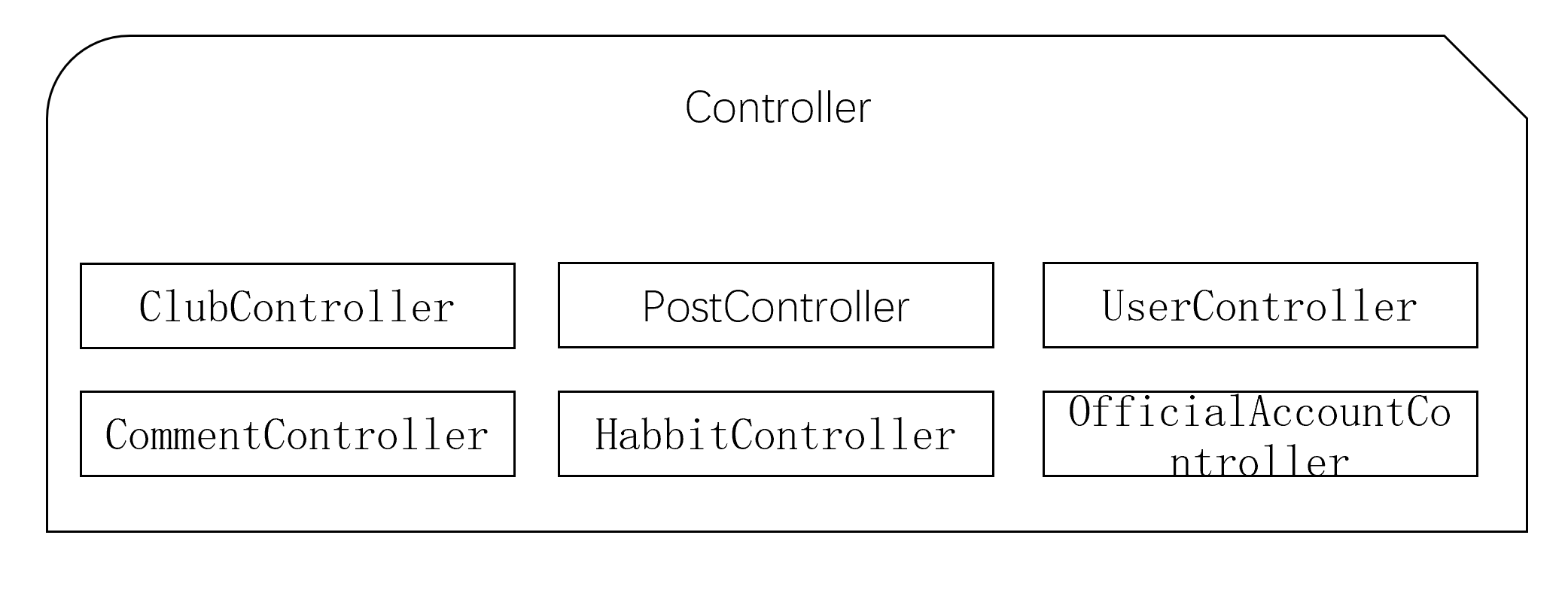
## 在构架方面具有重要意义的设计包

[对于每个重要的包，都用一个小节来加以说明，其中应包括该包的名称、简要说明以及显示该包中所有重要的类和包的图。

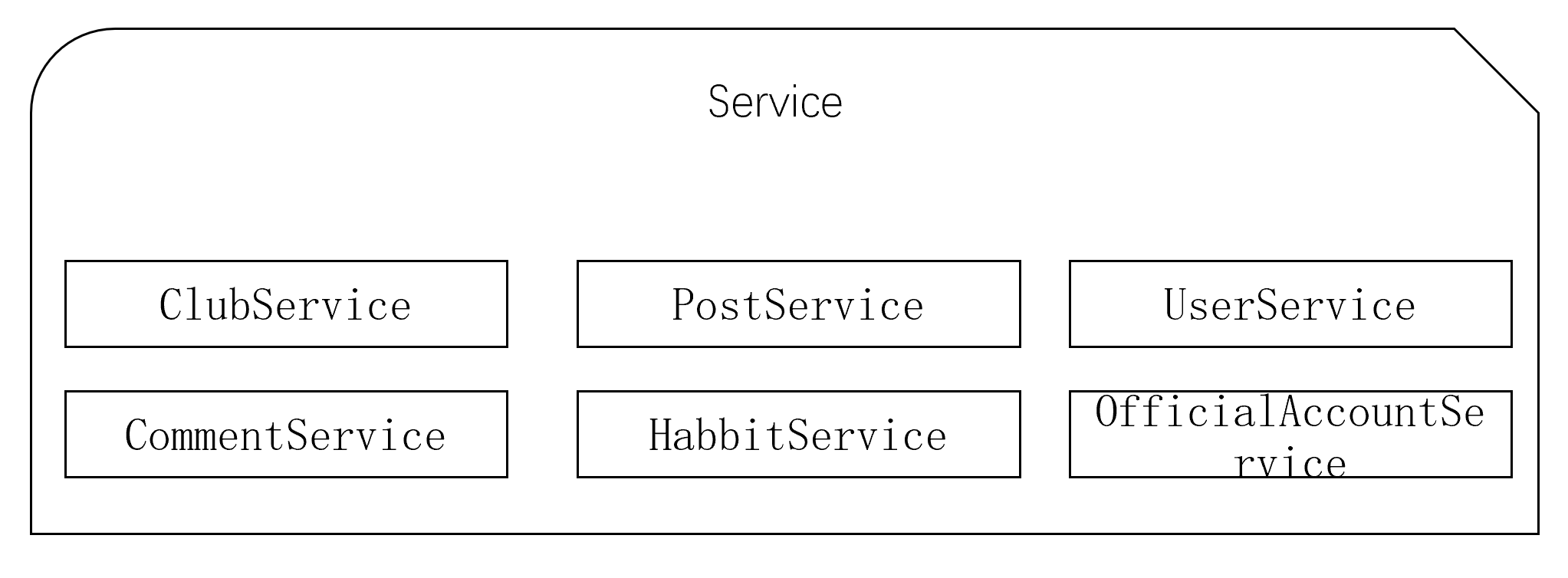
对于该包中的每个重要类，应包括其名称、简要说明，还可选择包括对其部分主要职责、操作和属性的说明。]

1. 交互层：static\_pages包：包含所有web静态页面和渲染素材，由vue框架实现。
2. 控制层：controller包：包含响应给网页端发送的HTTP请求。包含的类：

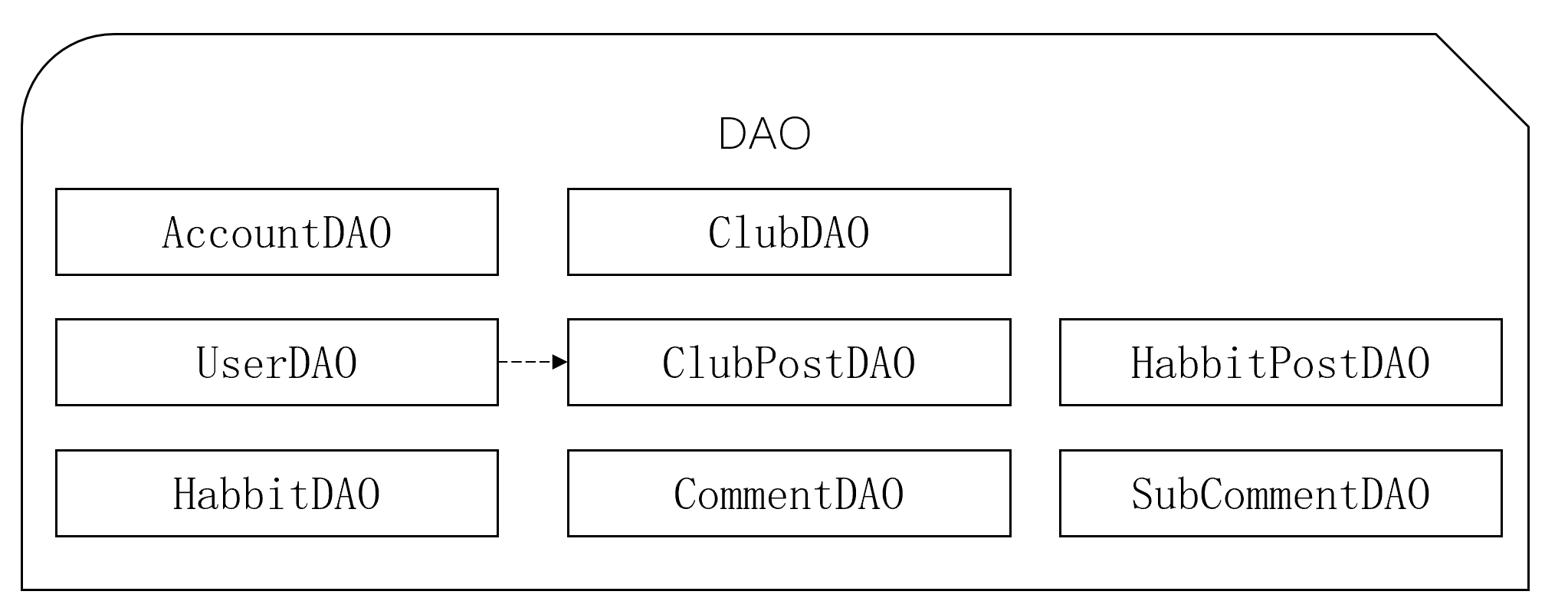
ClubController PostController UserController CommentController HabbitController OfficialAccountController



1. 业务层：Service包，包含使用Springframework框架实现的Service类，主要实现业务逻辑。

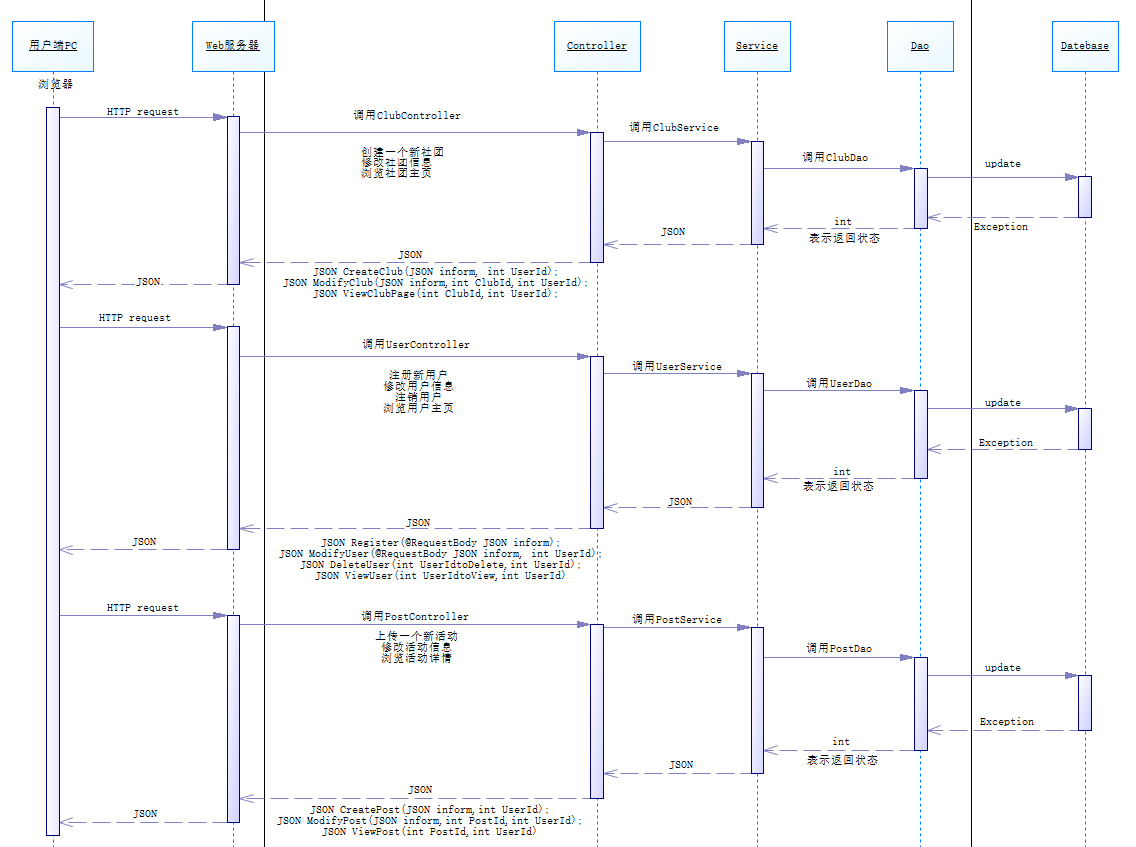


1. 持久层：DAO包，使用Springframework中的Repository框架实现的数据库连接和访问。



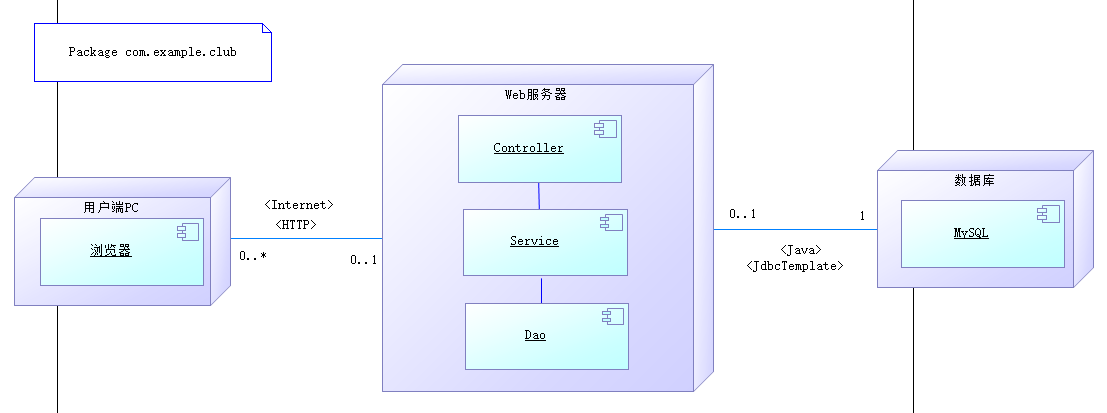
# 进程视图

[本节说明将系统分解为轻量级进程（单个控制线程）和重量级进程（成组的轻量级进程）的情况。本节的内容按照各个通信或交互的进程组来进行组织。说明进程之间的主要通信模式，例如消息传递、中断和会合。]



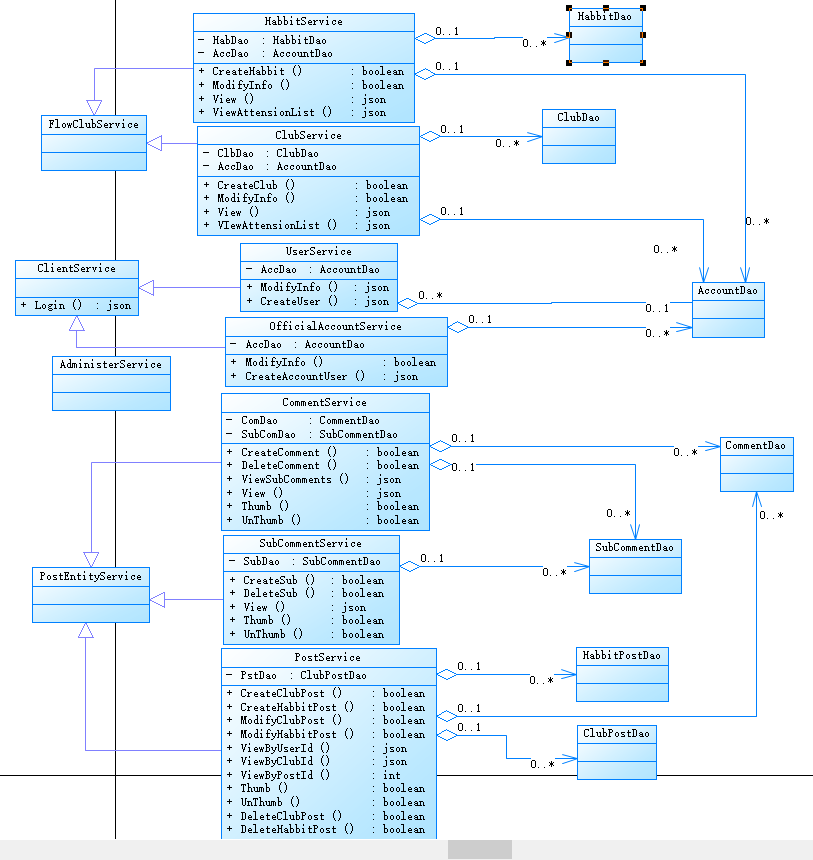
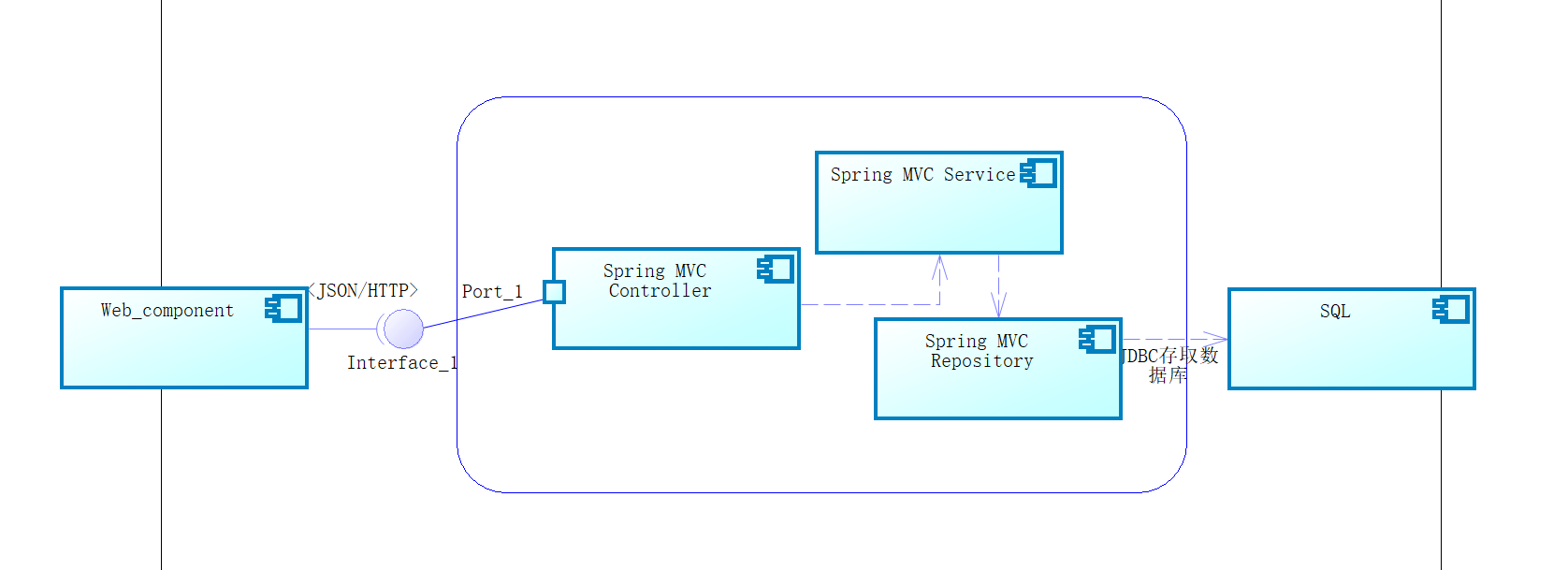
# 部署视图

[本节说明用来部署和运行该软件的一种或多种物理网络（硬件）配置。对于每种配置，它至少应该指出执行该软件的物理节点（计算机、CPU）及其互连情况（总线连接、LAN 连接、点到点连接等）。另外还要包括**进程视图**中的各进程到物理节点的映射。]



# 实现视图

[本节说明实现模型的整体结构、软件分解为实现模型中的层和子系统的情况，以及所有在构架方面具有重要意义的构件。]



# 技术视图

[技术栈的选型，包括编程语言、开发工具、框架、数据库、中间件等。]

编程语言：html，JavaScript，css，Java

开发工具：Intellij IDEA ， VSCode

框架：Spring MVC， Bootstrap， JQuery.js ， VUE.js

数据库：MySQL

# 数据视图（可选）

[从永久性数据存储方面来对系统进行说明。如果几乎或根本没有永久性数据，或者设计模型与数据模型之间的转换并不重要，那么本节就为可选。]

[如果需要，可增加其他视图。]

# 核心算法设计（可选）

[对系统中的核心算法进行设计。如果没有什么重要的算法，那么本节就为可选。]

# 质量属性的设计

[说明软件架构如何促成诸如性能、可扩展性、可靠性、易用性、可移植性等所有系统能力（而非功能）的实现。如果这些特征具有特殊的意义（例如在安全性或保密性方面的意义），则应该对它们进行详细的说明。]

|  |  |
| --- | --- |
| 质量因素 | 设计战术 |
| 易用性 | 出错时，给用户提供错误返回类型，并及时返回到正常页面 |
| 可靠性 | 在后端和前端页面分别检查错误和异常，如用户登录异常和注册异常，内容获取异常，非法数据等。 |
| 可移植性 | 采用bootstrap框架下的响应式页面，在更改页面大小或者在不同浏览器下都能有自适应的页面呈现。 |