本 科 毕 业 设 计 （ 论 文 ）

基于 Ionic 架构的网络数据管理平台的设计与实现

Design and realization of network data management platform based on Ionic architecture

学　　院：软件学院

专　　业：软件工程

学生姓名：XXX

学　　号：XXX

指导教师：XXX

北京交通大学

2016 年 5 月

学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：

中文摘要

摘要：随着互联网的快速发展，当今社会已进入“大数据”时代，数据的运用、处理已经成为科研、教育等领域必不可少的一环。在数据管理的过程中，常规的数据库、表单等形式比较繁琐；现有的数据管理平台存在各种各样的问题，包括缺少用户协作编辑、用户操作过于复杂等。因此，开发一款全新的网络数据管理平台已是一件迫在眉睫的事情。

XXX 公司成立于 2015 年，公司的口号与目标是“用数据实现共产主义”。公司致力于开发一款安全、高性能、低成本的网络数据管理平台，该平台可以在线创建、维护数据库的系统，能够快速生成以及实时维护表单。本平台采用网页客户端的形式，分为库管理、目录管理、条目管理、空间管理、表单管理等八个模块。在本项目中，本人主要负责其中四个模块的开发,每个模块包括了前端和后台两个部分。前端主要是基于 ionic 架构的移动网页端，后台包括 JAVA 服务器、Solr 搜索引擎、关系型数据库和非关系型数据库等部分。开发过程遵循软件工程的规范流程，从需求分析到架构设计，再到模块设计与实现，最终完成本系统。

本项目现在已于 2016 年 4 月 1 日正式上线，主要功能已基本实现，目前已有数十家机构和组织注册并使用，用户反馈良好。同时，新功能的添加与测试、系统的维护与修改正在不断进行中。

关键词：数据、管理、表单

ABSTRACT

ABSTRACT: with the rapid development of the Internet, today's society has entered the era of "big data". The use of data processing has become an indispensable part of the field of scientific research, education and other fields. It's difficult to manage data by normal database,form and more,but there are many problems in current data management platform,include lack of users collaboration editing,complex of users operation. Therefore, it's a matter of urgent to develop a new network data management platform.

XXX company was founded in 2015, whose slogan and the goal is to "achieve communism with data".The company is committed to the development of a safe, high-performance, low-cost network data management platform, which can create and maintain the database system, quickly generate and real-time maintenance of the form. The platform adopts the form of web client, which is divided into eight modules: library management, directory management, entry management, space management, form management, etc.. In this project, I'm mainly responsible for the development of four modules, each module includes two parts of the front and back. Front end is mobile Web terminal which is mainly based on the ionic architecture, and the background includes JAVA server, Solr search engine, relational database and non relational database, etc.. The development process follows the software engineering standard process, which is from the demand analysis to the architecture design, and then to the module design and implementation, ultimately complete the system.

The project has been formally launched in April 1, 2016, the main function has been basically realized, there are dozens of organizations to register and use, the user feedback is nice. New function of the addition and testing, system maintenance and modification are ongoing.

KEYWORDS: data, management, form

目　　录

1 绪论

1.1 课题研究背景

随着互联网的快速发展，正如全球知名的咨询公司麦肯锡所述，“数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于海量数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来。”[1]“大数据”在化学、物理学、生物科学等研究领域，以及金融、商贸等行业中的位置越发重要，这个世界已进入了“大数据时代”。

在这样的时代背景下，数据的运用、处理已经成为科研、教育等领域必不可少的一环。管理数据，比较常用的管理工具是各种数据库，例如 SQLserve ,Oracle，MySQL，但这些数据库都具有很多局限性，例如都要求用户熟练使用 sql 语言，在建表时需要考虑很多逻辑关系等。因此，拥有一个可以自由创建属于自己的表单，实时在线维护数据的系统，已是这个时代迫在眉睫的需求。

1.2 国内外研究现状

现阶段国内外在数据管理平台都有不少的尝试，但至今为止都尚未有一个在该领域拥有绝对统治力的系统，它的发展还不是很成熟。在调查的过程中，公司项目组研究并分析了以下几个系统。

首先是 wufoo，这是一个创建时间比较早的在线表单工具，用户可以使用它实现在线创建表单、表单嵌入网站或博客、Email 获取表单提醒、自定义表单主题等功能，其功能非常强大。它的优点在于，提供表单服务已持续 10 年，拥有丰富的经验，并且由于是国外的产品，在面向海外用户做数据搜索时，有明显的性能优势。但它也有不少缺点，如免费版只能创建 3 个表单、中文支持较差等方面，但最关键的是，它在用户间的数据共享方面提供的功能不足，无法实现协作编辑等功能。

其次是金数据，这是一个近几年上线的表单工具，可用来设计表单，制作在线问卷调查，组织聚会，询问意见，整理团队数据资料，获得产品反馈等。它的优点在于，它本身是国内的产品，表单编辑容易上手，使用体验较好，还支持表单复制功能。但它也有一些缺点，很重要的一点是它不能提供订制服务，这样不能满足一些对表单有特殊需求的用户的需要。

最后是 Mike CRM，它把自己定位成 CRM 工具与联系人管理工具。它同时具有编辑表单的功能以及联系人与邮件的服务。麦客主要的优点在于其统计功能，它能够提取受访

者信息到通讯录并分组，并从表单和数据统计两方面开始客户信息管理。它的优点在于，轻量简洁，并且支持移动端创建表单；而它的缺点则是缺少模板，用户在创建表单的过程中，所有的设计都需要自己去完成。

综上，项目组总结后得出，现在的数据管理平台的缺点主要集中在这几个方面：

1. 没有考虑到用户的协作编辑功能；
2. 没有提供表单的模板；
3. 没有订制表单的服务。

因此，在本公司的数据管理平台的过程中，这三个问题是需要重点考虑的。

1.3 课题研究内容及意义

为了解决上述存在的各类问题，方便各行各业的用户简单、快捷地管理、分享数据，本公司决定开发一种全新的网络数据管理平台——“数据酷”。“数据酷”是一个可以在线创建、维护数据库的系统，面向的是编程零基础的企业、机构及个人用户，能够快速生成以及实时维护表单，提供安全、高性能、低成本的数据库服务。同时，“数据酷”能提供共享数据库、场景模板库和定制化服务。共享数据库可以浏览、搜索各行业的公开领域的专业数据，也能够分享自己数据，还能让各个用户共同编辑、维护所属组织内的数据，达到协作编辑的目的；场景模板库可以选择使用来自各行业的数据库场景模板，并在此基础上实现自己的个性化需求；定制化服务能够设计、创建属于用户自己的个性数据库。

1.4 个人主要工作

“数据酷”网络数据管理平台系统主要分为了八个模块，分别是库管理模块、目录管理模块、条目管理模块、空间管理模块、表单管理模块、模板管理模块、权限管理模块、社区管理模块，其中本人主要负责空间管理模块、表单管理模块、权限管理模块和社区管理模块这四个模块，也参与了其他四个模块的部分开发，本文对我主要负责的四个模块进行详细描述，对其功能简述如下：

（1） 空间管理模块：该模块主要是对用户空间的管理，包括我创建的、我编辑的、我浏览的、共享给我的四个模块，每个模块分别包含对库、目录、条目的处理。

（2） 表单管理模块：该模块主要是对用户表单的管理，包含创建、编辑、删除表单三个模块，其中表单编辑模块又分为了表单编辑和字段验证两个功能。

（3） 权限管理模块：该模块主要是对用户权限的管理，包含浏览权限、编辑权限、新建权限等。

（4） 社区管理模块：该模块主要是对用户社区的管理，包括评论、点赞、通知、论坛四个模块。

1.5 论文组织结构

本论文是以项目“‘数据酷’网络数据管理平台”的设计与实现作为研究对象，依据软件工程项目开发的流程进行论述，主要分为五个模块，绪论、需求分析、系统架构、模块设计与实现、总结。

第一章：绪论，分为五个小节，前三个小节对课题的研究背景、国内外研究现状和研究内容及意义进行简述，表明了本课题的研究背景和需要解决的问题；后两个小节说明了个人的主要工作和论文的组织结构。

第二章：需求分析，分为需求概述、功能需求、非功能性需求三个小节，其中需求概述介绍了整个项目的需求；功能需求介绍了我主要负责的四个功能模块的需求。

第三章：系统架构，分为整体架构、功能架构、开发环境、关键技术四个小节，其中整体架构介绍了项目的整体框架；功能架构介绍了项目各个功能的框架；开发环境介绍了项目开发使用的平台、软件等工具；关键技术介绍了项目使用的主要技术和架构。

第四章：模块设计与实现，分为四个小节，分别介绍了我主要负责的四个功能模块的设计与实现过程，主要包括设计描述、模块流程图、模块系统顺序图、类图和界面设计。

第五章：总结，对论文撰写的总结和对所有支持、帮助我完成这个项目的人表示感谢，以及参考文献和英文文献翻译。

2 需求分析

“‘数据酷’网络数据管理平台”是一个提供在线数据管理的系统，主要功能是让用户简单、便捷地创建表单来存储数据，并可以将自己的数据分享给其他人，也能与其他人协作编辑数据。它的主要功能还包括利用模板创建库、用户空间、用户社区等。

2.1 需求概述

2.1.1 项目概述

正如前文所述，现如今已出现了不少网络数据管理系统，除了前文提到的 wufoo、金数据、Mike CRM，还有 Google Form、简道云等。综合来看，网络数据管理系统一般都有如下几个功能：

（1） 数据存储：最基本的功能，提供给用户数据存储空间。

（2） 表单编辑：前端的主要功能，提供可视化的数据库编辑界面。

（3） 数据管理：基本功能，提供数据管理的功能。

（4） 用户中心：基本功能，包括登陆、注册、消息等。

但是，在调研的过程中，公司项目组发现了过去的网络数据管理系统存在一些功能的缺失，这就是前文所述的缺点，下面是对各个缺点的分析。

（1） 没有考虑到用户的协作编辑功能。这是一个很重要的方面，能够允许用户与朋友、同事、授权的用户同时对数据进行编辑，免除了团队中数据整理、汇总的麻烦，因此这是需要考虑。

（2） 没有提供表单的模板。对于很多用户，即使是提供了一个通过点击按钮和输入信息就可以创建数据空间，但花费 1 到 2 个小时来制作一个表单，依然是比较麻烦的事情。尤其是对于存储数据比较少的公司和私人用户，他们更愿意选择直接使用一个简单的模板来创建自己的数据空间。

（3） 没有订制表单的服务。如果用户不能找到合适的模板来创建自己的数据空间，但又觉得去学习编辑表单比较麻烦，毕竟表单编辑一般只需要用户操作一次，为此去学习操作方法比较浪费时间；这个时候就需要系统提供订制表单的服务，用户在提交需求之后，由工作人员为用户建立满足用户需求的表单。

综上，项目组确定了“‘数据酷’网络数据管理平台”应具有的功能。同时，为了更加方便用户进行数据分享、交流，项目组决定加入社区功能，使用户可以相互评价、点赞各自的数据，因此，本平台就分为了以下八个方面的功能：数据存储、表单编辑、数据管理、用户中心、协作编辑、使用模板、订制表单、用户社区。下面是对各个功能的简单介绍：

（1） 数据存储：数据管理系统的基本功能，提供数据存储的空间，用户可以将自己的数据保存在云服务器，并在需要时对其进行读取、修改、删除等操作。数据存储采用关系型数据库（RDS）和非关系型数据库（NoSQL）结合存储的方式，RDS 存储数据间的关系，NoSQL 存储数据的具体信息。存储类型包括文字、图片、音频、视频等。

（2） 表单编辑：向用户提供可视化的表单编辑页面，用户可以通过点击按钮、拖动框体等直观的形式编辑自己的表单，免除了 SQL 语句的输入，简化用户的操作。

（3） 数据管理：每一位用户都拥有自己的数据空间，在空间中可以对自己的数据进行管理。数据的操作需要权限的支持，如果用户对一个资源具有编辑权限，即可对其进行修改并保存。权限依赖于资源本身，而不是用户本身。

（4） 协作编辑：多个用户可以对同一资源进行操作，方便数据的协同管理，让用户能告别繁琐的 EXCEL 上传、下载、发送，使同事、朋友之间的信息传递更加简单便捷。

（5） 使用模板：如果用户不喜欢花太多的时间去编辑表单，系统可以提供大量的表单模板，用户只需检索到适合自己使用的表单类型，然后直接在此基础上进行一些简单的修改，即可得到自己需要的表单。

（6） 订制表单：对于有特殊需求的用户，可以通过系统发送请求来订制表单，后台服务人员在接收到用户请求之后即可根据需求帮助用户建立合适的表单，提供给用户使用。

（7） 用户社区：为了方便用户间的交流，系统提供了一个额外的服务——用户社区。用户可以对他人的库进行评价、点赞，也可以创建一个话题，在话题中对库的信息与他人进行讨论。

2.1.2 需求概述

在确定了用户的需求之后，首先要考虑在技术层面如何实现这个系统，其中最基本的是数据的结构和存储。

由于这个系统给用户展现的应该是一个比较直观的“数据库”，故而“库”应该是最基本的概念。用户在使用这个系统时，首先应该创建一个库，并定义存储数据的类型和数据之间的关系，然后再将数据存放到这个库中。

在拥有了库这个容器之后，剩下的就是存储数据。项目组决定将每一条数据称为一个“条目”，每一个条目可以包含很多个字段，例如文本、图片、音频、视频等等，用户可以根据需要自行设计。字段可以理解为类似数据库中一个表的属性。

如果把库比作一本书，那么条目就是书中的文字。但众所周知，如果一本书只有封

面和文字，对其内容的检索和分类是非常麻烦的。因此，在确定了“‘条目’存储于‘库’”这个基本关系的前提下，项目组又引入了一个新的概念——“目录”。就像书的目录一样，“目录”存储在库中，同时能够把“条目”放到不同的目录下，也即是每一个条目可以在逻辑上属于不同的目录，这样可以方便用户检索条目，也允许用户使用多种方式对条目进行分类。

下图是对数据的结构和存储的简单解释：

图 2-1 数据存储关系图

其中实线代表物理上的“属于”关系，虚线代表逻辑上的“属于”关系。

综合以上调查、分析结果，为便于系统开发以及维护，项目组将这八个功能进行了整合、分离，最终确定了绪论中所述的八个模块：

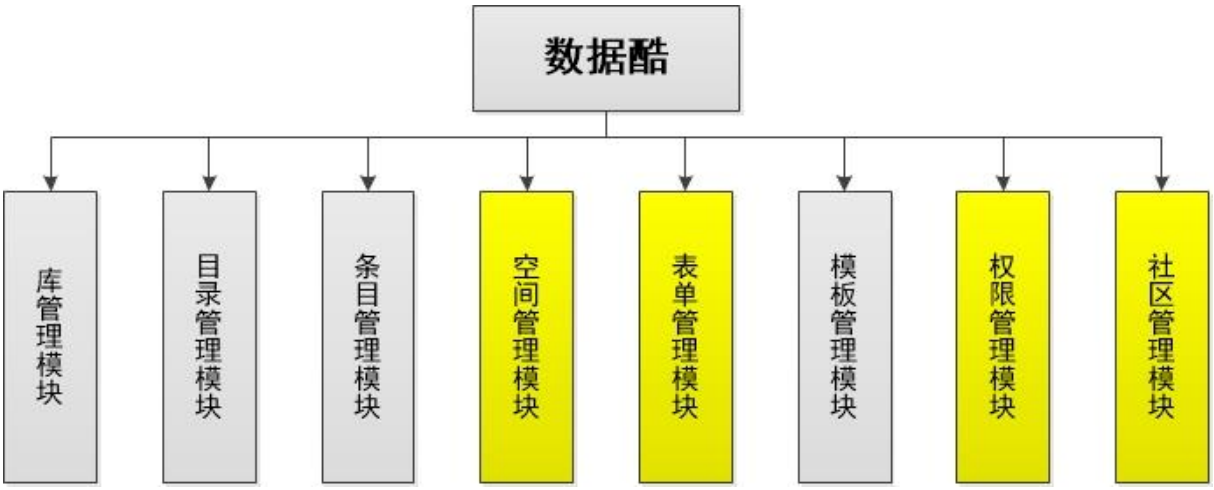


图 2-2 系统模块图

其中标记为黄色的空间管理模块、表单管理模块、权限管理模块和社区管理模块为本人主要负责的模块，其详细需求将在功能需求中详细描述。

2.2 功能需求

2.2.1 空间管理模块

（1） 功能描述

本模块主要是对用户空间的管理。

首先，用户空间是每一位用户在除了个人信息外还对应的一组信息，它以个人名义进行的操作及所操作的资源（包括库、目录和条目三类资源）,都属于这个空间，也可以理解为在个人空间中进行的操作。空间需要记录并展示这些操作及资源。其详细的需求内容如下：

表 2-1 空间内容分类表

个人空间下目录名称	展示内容	说明
我创建的	按创建时间倒序展示用户以个人名义创建的库、目录、条目	三类资源不分类，混合显示
我编辑的	按编辑时间倒序展示用户以个人名义编辑的库、目录、条目	三类资源不分类，混合显示。同一个资源，多次编辑，只显示最新的一条记录
我浏览的	按浏览时间倒序展示用户以个人名义编辑的库、目录、条目	三类资源不分类，混合显示。同一个资源，多次浏览，只显示最新的一条记录
共享给我的	按共享给我这个操作发生时间倒序，展示所有将该用户设置为编辑人、浏览人、综合管理、所有者（该部分见权限管理模块的详细设计）的资源，以库、目录、条目的形式显示，每个资源只显示一条，不需显示执行这个操作的人及所设置的权限等详细信息	各类资源,各类角色不分类,混合显示。同一资源，应只显示一条记录
全部	按时间倒序，展示以上四类资源	各类资源，混合显示，同一个资源只显示一次,无论它在空间中有几条操作记录

其次，个人空间拥有权限，该权限作用于个人空间本身，在个人空间中 创建的库继承空间的权限，用户可以修改这个权限。

最后，在空间下可以创建库，同时每个目录下显示在目录下包含的资源（或日志）的数量。

（2） 用例图

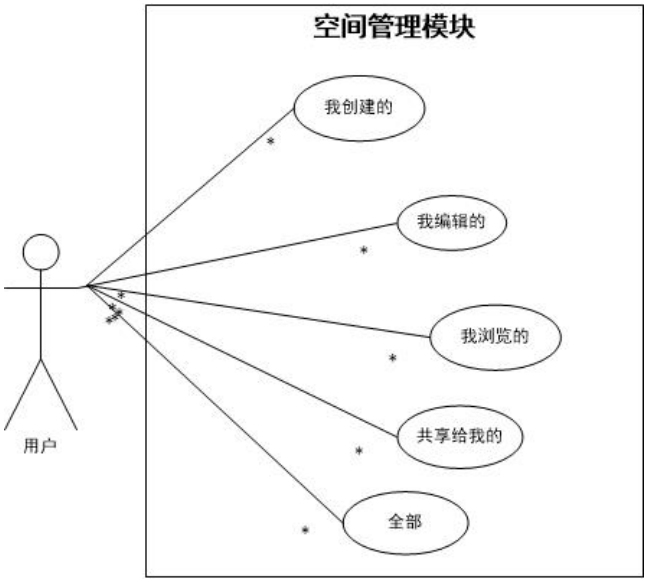


图 2-3 空间管理模块用例图

（3） 用例描述

1） “我创建的”用例

1） 系统收到打开“我创建的”空间信息显示请求调用服务器业务处理逻辑。

2） 系统为用户显示“我创建的”空间信息页面

3） 系统显示从后台获取的与“我创建的”相关的空间信息列表

2） “我编辑的”用例

1） 系统收到打开“我编辑的”空间信息显示请求调用服务器业务处理逻辑。

2） 系统为用户显示“我编辑的”空间信息页面

3） 系统显示从后台获取的与“我编辑的”相关的空间信息列表

3） “我浏览的”用例

1） 系统收到打开“我浏览的”空间信息显示请求调用服务器业务处理逻辑。

2） 系统为用户显示“我浏览的”空间信息页面

3） 系统显示从后台获取的与“我浏览的”相关的空间信息列表

4） “共享给我的”用例

1） 系统收到打开“共享给我的”空间信息显示请求调用服务器业务处理逻辑。

2） 系统为用户显示“共享给我的”空间信息页面

3） 系统显示从后台获取的与“共享给我的”相关的空间信息列表

5） “全部”用例

1） 系统收到打开“全部”空间信息显示请求调用服务器业务处理逻辑。

2） 系统为用户显示“全部”空间信息页面

3） 系统显示从后台获取的与“全部”相关的空间信息列表

2.2.2 表单管理模块

（1） 功能描述

本模块主要是对表单的管理。

表单是一个非常重要的功能，它允许用户根据自己的喜好，创建一个表单，在表单中能根据需要添加字段，主要字段如下：

表 2-2 表单字段信息表

字段名	中文名	根类型
name	名称	text
text	单行文本框	meta
textarea	多行纯文本框	text
number	数字	meta
combo	组合项	meta
toggle	开关	boolean
link	链接	meta
radio	单选	list
checkbox	多选	list
selectbox_local	本地下拉框	list
multi_selectbox_local	多级本地下拉框	combo
value_ref	条目引用输入框	refid
selectbox_ref	引用下拉框	refid
nickname	别名	text

表 2-2<续> 表单字段信息表

username	用户名	text
doc_ref	表引用	refid
doc_img	头像	link
file	文件	link
image	图片	link
video	视频	link
audio	音频	link
document	文档类文件	link
tag	分类标签	tag
phone_number	电话	combo
email	电子邮箱	combo
password	密码	text

更多字段将在未来根据用户需求逐步添加。

每一个字段应当有其限制条件，当前的限制条件有：

表 2-3 字段限制条件表

英文名	中文描述	类型	填写限制
name	字段名（始终显示）	text	字符数 1-20
field_name_edit	字段名（只在编辑模式下显示）	text	字符数 1-20
field_name_view	字段名（只在浏览模式下显示）	text	字符数 1-20
description	字段下方的说明文字（始终显示）	textarea	字符数 1-100
description_edit	字段下方的说明文字（只在编辑模式下显示）	textarea	字符数 1-100
description_view	字段下方的说明文字（只在浏览模式下显示）	textarea	字符数 1-100
info	字段右边圆圈问号标志里面的内容，该内容为富文本（始终显示）	textarea	无限制
default_value	默认值，用户不操作也保存的值	text	无限制

表 2-3<续> 字段限制条件表

info_edit	字段右边圆圈问号标志里面的内容，该内容为富文本（只在编辑模式下显示）	textarea	无限制
info_view	字段右边圆圈问号标志里面的内容，该内容为富文本（只在浏览模式下显示）	textarea	无限制
placeholder	字段内显示的提示信息，点入后消失，只在编辑模式展示	textarea	无限制
assist_fill	用户点入时自动出现的内容，例如点入网址，前面默认出现【http://】，只在编辑模式展示	text	无限制
prefix	前缀（始终显示）	text	字符数 1-10
prefix_edit	前缀（只在编辑模式下显示）	text	字符数 1-10
prefix_view	前缀（只在浏览模式下显示）	text	字符数 1-10
suffix	后缀（始终显示）	text	字符数 1-10
suffix_edit	后缀（只在编辑模式下显示）	text	字符数 1-10
suffix_view	后缀（只在浏览模式下显示）	text	字符数 1-10

（2） 用例图

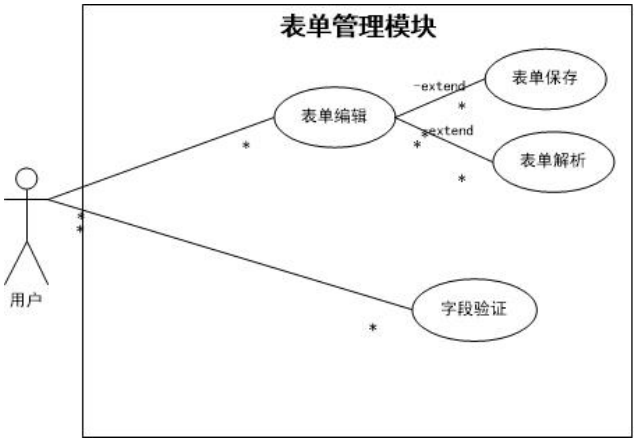


图 2-4 表单管理模块用例图

（3） 用例描述

（1） 表单编辑用例

1） 用户点击“编辑表单模板”选项，将编辑请求发送到后台

2） 后台创建新表单，同时前端显示表单编辑界面

3） 用户编辑表单，完毕后点击保存

4） 后台将表单信息保存

（2） 字段验证用例

1） 用户点击需要添加字段验证的表单字段，将请求发送到后台

2） 读取字段的信息并发送到前端，前端显示字段设置界面

3） 用户设置字段限制条件，完毕后点击保存

4） 后台将字段限制信息保存

2.2.3 权限管理模块

（1） 功能描述

本模块主要是对用户权限的管理。

权限模块用来限制用户对指定资源（包括库、目录、条目）的操作权限，权限被定义在资源上而不是基于用户的角色。权限的定义被储存在权限表中，通过二维关系与指定资源进行绑定。权限从低到高，分为以下几个等级：

1） 不可见 (na)

对用户来讲，等于不存在（虚拟权限，不可选）

2） 摘要可见 (summary)

可以看见资源所有位置的摘要信息。

3） 详情可见 (view)

可以看见该资源的详情，并拥有该资源的入口。

4） 创建 (create)

可以新建指定资源的下级资源。

5） 编辑 (edit)

可以更新内容，如果没有开启审核功能立即生效，否则提交给有审核权限的人审核通过后才生效。包括增删改查（删除只包括内容的删除，不包括资源本身的删除）操作。

6） 编辑审核 (approve)

更新内容需要审核通过后才能生效。

7） 管理员 (admin)

具有之前所有的权限，并且可以编辑这些的权限。

8） 所有者 (superAdmin)

拥有所有权限，并且可以指定管理员和分配权限，可对资源进行删 除操作。每个资源最少有一个所有者。

权限管理模块包含权限应用，权限编辑两个用例。

（2） 用例图

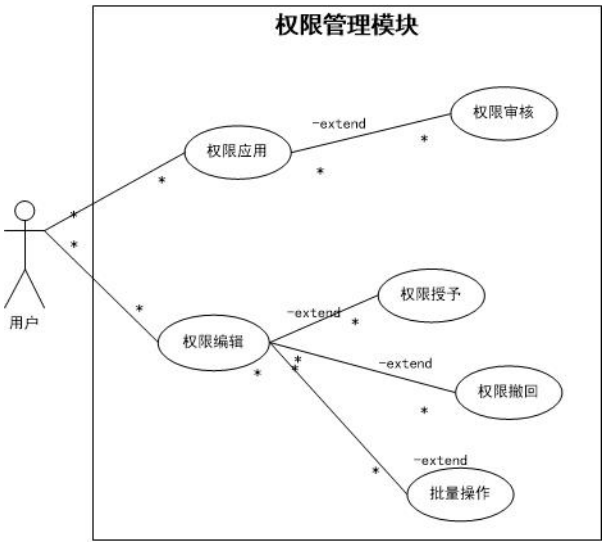


图 2-5 权限管理模块用例图

（3） 用例描述

（1） 权限应用

1） 在对所有资源的操作中，前端都将用户信息及资源信息发送到后台

2） 后台从权限表中获取二者的关系，判断操作是否通过，并将结果发送到前端

3） 前端根据后台结果显示

（2） 权限编辑

1） 具有“权限编辑”权限的用户可以在前端修改资源的权限

2） 用户点击保存，将信息发送到后台

3） 后台判断权限修改是否合法，若合法则存储

2.2.4 社区管理模块

（1） 功能描述

本模块主要是对用户社区的管理，包括评论、点赞、通知、论坛四个模块，功能如下：

1）评论：对一个库、条目等进行评论，评论可以提到用户。

2）点赞：对一个库、条目、评论等进行点赞。

3）通知：包括用户私信和系统通知两个功能。

4）论坛：可以自由创建话题，话题分为两类，基于库、条目的话题和不基于任何内容的话题。每一个话题都类似于一个贴子，可以在其中自由评价、点赞。

（2） 用例图

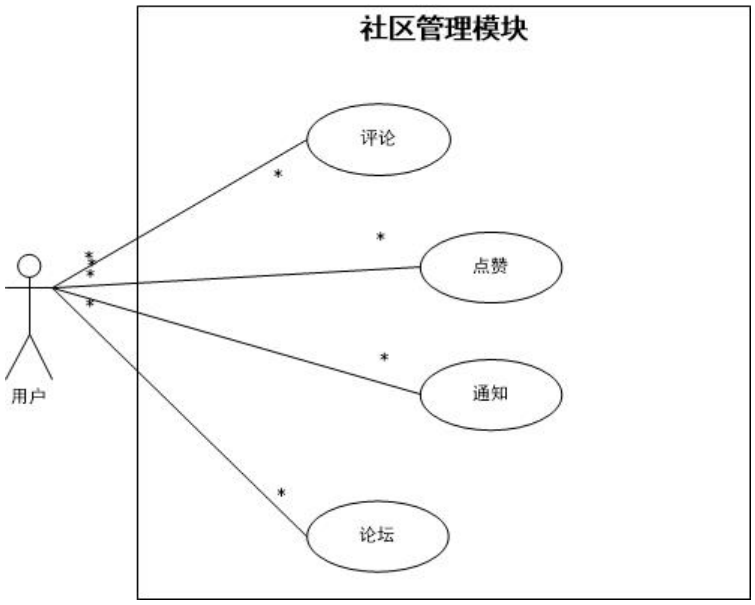


图 2-6 社区管理模块用例图

（3） 用例描述

（1） 评论

1）在一个库的详情页面下，用户可以输入文字进行评论

2）点击评论按钮后，前端将用户的评论内容发送到后台

3）后台建立评论与用户的关系，并存储评论内容

（2） 点赞

1）在一个库的详情页面下，用户可以点击点赞按钮

2）点击点赞按钮后，前端将用户点赞这条信息发送到后台

3）后台建立点赞与用户的关系并保存

（3） 通知

1） 用户可以给其他用户发送消息

2） 消息保存在后台，由两个用户的 ID 建立关系

3） 用户也可以接收到系统消息

（4） 论坛

1） 用户可以创建一个话题

2） 点击创建按钮后，前端会将用户 ID 发送到后台

3） 后台存储用户信息以及话题内容

3 系统架构

3.1 整体架构

“‘数据酷’网络数据管理平台”系统设计基于 B/S 架构，全称为 Browser/Server，即浏览器/服务器结构。

Browser 指的是 web 浏览器，将其作为主要的展现层，并实现极少数的事务逻辑。在本系统中，为了方便用户的使用，顺应当下移动端迅猛发展的潮流，公司决定首先开发为移动网页端，然后再根据需要开发 PC 网页端。

Server 指的是后台服务器，主要实现数据的存储、访问以及逻辑处理。本系统采用 JAVA 服务器，将服务器分为了三个部分，分别是数据存储层、数据访问层和业务逻辑层。同时，为了提高搜索的效率，项目组决定加入 Solr 这个独立的企业级搜索应用服务器。

系统架构如下图所示：

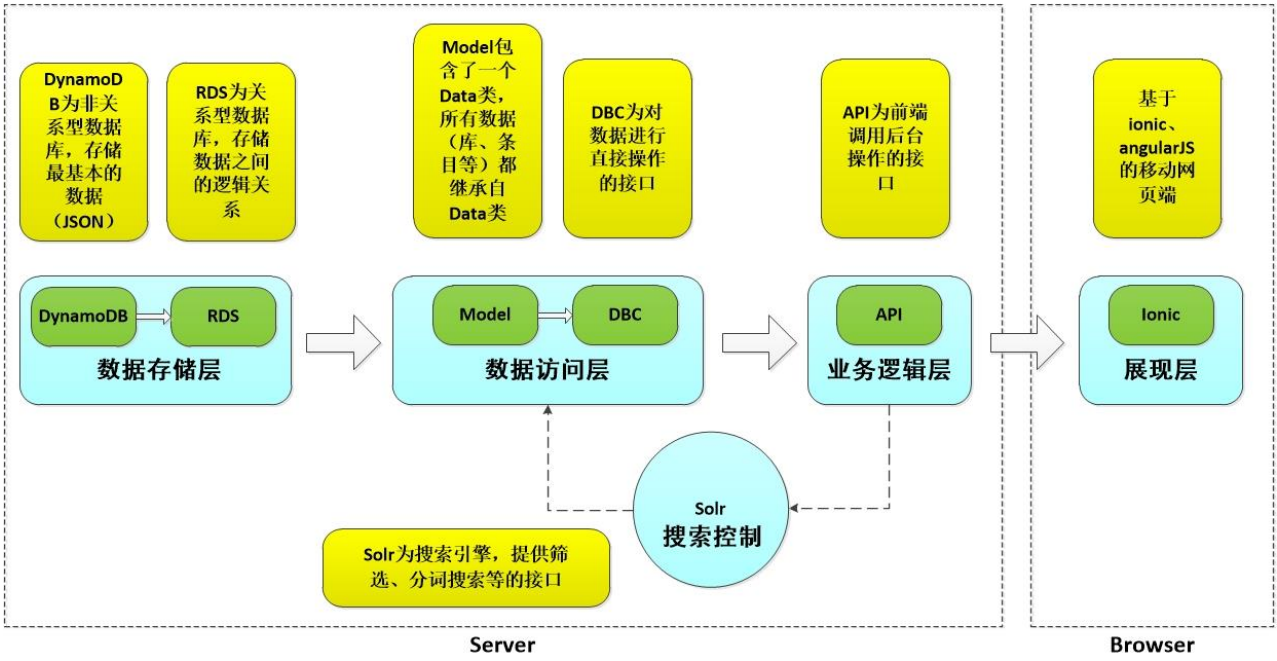


图 3-1 系统技术架构设计图

3.2 功能架构

本节将分别介绍上一节所述各个部分的架构。

（1） 数据存储层

这一层为系统的最底层，主要功能为存储用户的数据。

通过需求分析可知，项目组将用户数据的结构分为了库、目录和条目三个层次，并且它们两两之间都存在关系。因此，项目不能采用传统的关系型数据库存储方式，而是

使用了关系型数据库（RDS）和非关系型数据库（NoSQL）一起对数据进行存储。

首先，将数据之间的二维关系存储在 RDS 中，包括表单、条目、目录、库等，主要使用的编码如下：

表 3-1 数据二维关系表

数据名	中文名	编码	说明
form	表单	11	用户编辑的库、 条目等的结构
doc	条目	12	一条数据
dir	目录	13	库下的结构，可以从逻辑上连接条目
space	空间	14	每个用户拥有的自己的空间
user	用户	15	一个用户
org	组织	16	多个用户共同属于的组织
base	库	17	用户建立的数据存储空间
permission	权限	18	用户的权限

主要的二维关系解释如下：

1) 和表单相关的关系

11xx00：表单和 xx 资源的关系，表示该表单用来解析条目 xx, xx 包括 12,13,14,15,16,17,18

111701：表单和库的特殊关系，表示该表单用来新建和编辑该库的条目

2) 和库相关的关系

17xx00：库和 xx 资源的关系，表示该资源属于该库,xx 包括 12,13

171301： 库和其下一级目录子目录的关系，表示该目录属于这个库，但不是库的一级子目录

3) 和目录相关的关系

131300：目录的上下级关系，表示目录 2 是目录 1 的子目录

131200：目录和条目的关系，表示该条目属于这个目录

131700：目录和库的关系，在目录是空间中的目录的时候，表示该库属于这个目录

4) 和权限相关的关系

18xx00：权限和 xx 资源的关系，表示该资源的权限用该权限来定义，xx 包括 12,13,14,16,17

5) 和用户，组织相关的关系

161500：用户和组织的关系，表示该用户属于这个组织

6) 和空间相关的关系

141700：空间和库的关系，表示这个库属于这个空间

数据之间的关系存储于 RDS 的二维关系表中，需要对数据进行操作时， 可以通过它查找数据的 id，然后从 NoSQL 中获取到具体的数据。

NoSQL 中数据的存储方式比较简单，只存在 key-value 和具体数据，类型如下：

表 3-2 非关系型数据库存储关系表

名称	类型	说明
key-value	字符串	id
data	json	具体数据

以上是数据存储的方式。

（2） 数据访问层

这一层的主要功能为对数据的处理，分为 Model 和 DBC 两个部分。

首先，在 Model 部分，由数据存储层的分析可知，本系统的所有数据存 储都可以由用户来设计，因此，在 Model 中设置了一个 Data 类，所有数据（库、条目等）都继承自 Data 类，这样方便数据的读取与解析。

其次，DBC 为对数据进行直接操作的接口，所有的数据操作都在 DBC 中完成，包括增、删、改、查这些基本功能以及如索引、数据关系处理等扩展功能。

（3） 业务逻辑层

这一层的主要功能为处理前端与后台之间的信息，包括 API 一个部分。API 为前端调用后台的接口，主要用来获取消息、调用后端接口等。该部分功能比较常规，故不作赘述。

（4） 搜索控制

这一层的主要功能是提供搜索的接口，包括 Solr 一个部分。

Solr 允许用户通过 http 请求，向搜索引擎服务器提交特定格式的 XML 文件，生成索引[11]；也可以通过 Http 的 Get 操作提出查找请求，并得到 XML 格式的返回结果[11]。

（5） 展现层

这一层为前端，主要功能为提供移动网页端的可视化操作界面。

前端分为四个主要功能模块，分别是发现、数据、社区、我。

发现：主界面，提供了热门数据库和热门条目，以及场景模板和维基数据库，还包括了外部应用的链接。

数据：用户的数据空间，包括共享给我的数据、我创建的数据、我编辑的数据和我浏览的数据四个记录内容。

社区：用户社区，主要包括赞、评论和消息。

我：用户信息管理，包括登陆、注册、找回密码等。

主界面如下图所示：

图 3-2 主界面展示图

对于各个部分所用的关键技术，将在 3.4 节详细介绍。

3.3 开发环境

3.3.1 硬件开发环境

CPU：i5 处理器，1.70GHz*4
内存：8G
硬盘空间：1TB

3.3.2 软件开发环境

操作系统：Linux
IDE: MyEclipse，sublime
版本控制工具：github
原型设计工具：Axure RP
包管理工具：npm
服务器：tomcat
数据库：DynamoDB、mysql
缓存：Redis

3.4 关键技术

3.4.1 Ionic 框架

本系统前端采用 Ionic 框架。Ionic 是一个基于 HTML5,用来开发移动 web 端的框架。它基于 PhoneGap 编译平台，绑定了 AngularJS 和 Sass，是一款非常具有潜力的前端框架。它的优点主要有：

（1） 性能出色：它操作的 DOM 较少，采用硬件加速，并且是非 jQuery 的框架，这些因素是其运行非常流畅。
（2） 融合 AngularJS：Ionic 可以说是移动端的 AngularJS，而 AngularJS 是现在比较主流的开发框架，有利于系统未来的维护。
（3） 兼容性强：同时兼容 iOS 和 Android。

3.4.2 AngularJS 框架

AngularJS 是 Ionic 的基础，故而本系统也使用了 AngularJS 框架。AngularJS 是一个为动态 Web 应用设计的结构框架，它扩展了 HTML 的语法，让使用者更清楚、简洁地

构建应用组件。它具有以下特点：

（1） 数据绑定：将 view 层的数据和 model 层的数据双向绑定，只要其中一方数据发生变化，另一方也会随之改变，简化了编码。

（2） 依赖注入：将后端语言的设计模式赋予前端代码，提高前端代码的重用性和灵活性。

（3） 代码模块化：将代码分为多个模块，每个模块由它的作用域，例如 controller、service、factory 等。

（4） directive：将功能封装为 HTML 的属性、注释或者 tag 等，增强了 HTML 的可阅读性。

3.4.3 Spring boot 框架

本系统后台采用 Spring boot 框架。Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化 Spring 应用的初始搭建以及开发过程[3]。该框架配置简单，方式固定，并且开发人员不再需要定义样板化的配置。Spring boot 主要应用于快速应用开发领域。其优点在于：

（1） 简洁：简化 Spring、Maven 及 Gradle 的配置，环境搭建速度快，学习门槛低。

（2） 方便：采用嵌入式 Tomcat，Jetty 容器，无需部署 WAR 包。

3.4.4 Solr

本系统采用 Solr 作为搜索引擎。Solr 是一个独立的企业级搜索应用服务器，采用 Java5 开发，基于 Lucene，支持全文搜索。它提供了层面搜索、命中醒目显示并且支持多种输出格式。它的优点有：

（1） 使用简单：易于安装和配置，提供了一个基于 HTTP 的管理界面。

（2） 稳定：很多大型网站都使用了 Solr，技术较为成熟。

（3） 兼容性强：Solr 包装并扩展了 Lucene，基本沿用了 Lucene 的术语，其创建的索引与 Lucene 搜索引擎库完全兼容。

3.4.5 DynamoDB

本系统底层数据存储采用非关系型数据库（NoSQL），最终选择了 Amazon 提供的 DynamoDB。它是一个完全托管的非关系型数据库服务，可以提供快速的、可预期的性能，也能实现无缝扩展。它的优点有：

（1） 可扩展性：因为基于键值对，数据之间没有耦合性，很容易水平扩展，同时

也支持在吞吐量和存储能力上无缝扩展。

（2） 性能好：DynamoDB 是基于键值对的，表中的主键和值相对应，读取不需要经过 SQL 层的解析，而且 DynamoDB 服务端的平均延迟通常只有几毫秒，性能非常高。

（3） 管理简单：不需要考虑硬件和软件的配置，也不需要考虑软件的安装和升级，只要你简单地创建一个数据库表，其他的事情由 AWS（Amazon Web Service）来完成。

3.4.6 Redis

本系统采用 Redis 来存储缓存数据。Redis 是一个使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型 Key-Value 数据库。它的特点有：

（1） 简易：简单灵活，上手快。

（2） 数据结构丰富：Redis 中值的类型不仅限于字符串，还支持如下抽象数据类型：字符串列表（list）、无序不重复的字符串集合（set）、有序不重复的字符串集合（sorted set）、键和值都为字符串的哈希表（hash）。

（3） 高速读写：由于 Redis 纯粹使用内存读写，故读写速度较高，这也导致它的存储成本高，适合于缓存这种数据量较少的场景。

4 模块设计与实现

4.1 空间管理模块

4.1.1 设计描述

空间管理模块的设计分为前端和后台两个方面。

首先是前端设计。前端主要负责展示及操作用户空间，分为两个模块，一个是展示信息的“我的数据”模块，另一个是目录模块。在我的数据模块中，前端将从后台获取的数据根据时间（包括创建、编辑、浏览等的时间记录）倒序显示，并根据数据的类型（库、目录、条目）分别使用不同的卡片模板。在目录模块中，用户可以选择全部、共享给我的、我创建的、编辑记录和浏览记录 5 类信息，在选择之后用户选择的信息类型将在“我的数据”模块中显示。

然后是后台设计。后台主要负责数据的存储及操作。数据存储包括两个部分，一个是关系型数据库，用来存储用户与数据的二维关系，包括用户 ID、数据 ID、操作时间、操作类型；另一个是非关系型数据库，用来存储数据，数据只以 ID 和 JSON 的形式存在。数据操作包括三个部分，一个是 DBC 模块，主要用来响应前端的请求，并进行反馈，这是主要响应前端的读取请求；一个是 API 模块，主要用来处理关系型和非关系型数据库中的数据，模块先从 DBC 模块获取到用户 ID 和数据 ID，然后从关系型数据库中获取到二者的操作类型和时间，再从非关系型数据中获取到数据的详细信息；最后一个是 Solr 搜索引擎，主要创建及存储索引，提供搜索和筛选的功能。

4.1.2 模块流程图

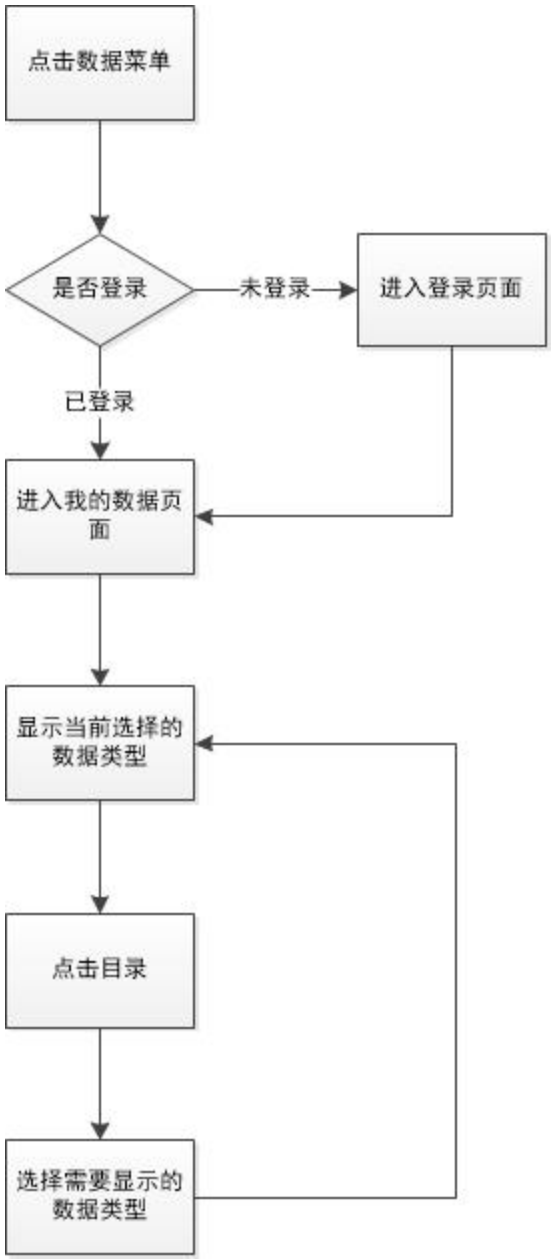


图 4-1 空间管理模块流程图

系统的流程如下：

用户点击数据菜单，根据登录状态判断是否进入我的数据页面。进入该页面后，前端会显示当前选中的数据类型（默认为全部数据）。如果用户点击其中的一条数据，则会进入该数据的详细信息。用户也可以点击目录按钮，选择其他数据类型，选中之后我的数据页面会刷新，显示用户选择的新的数据。

4.1.3 模块系统顺序图

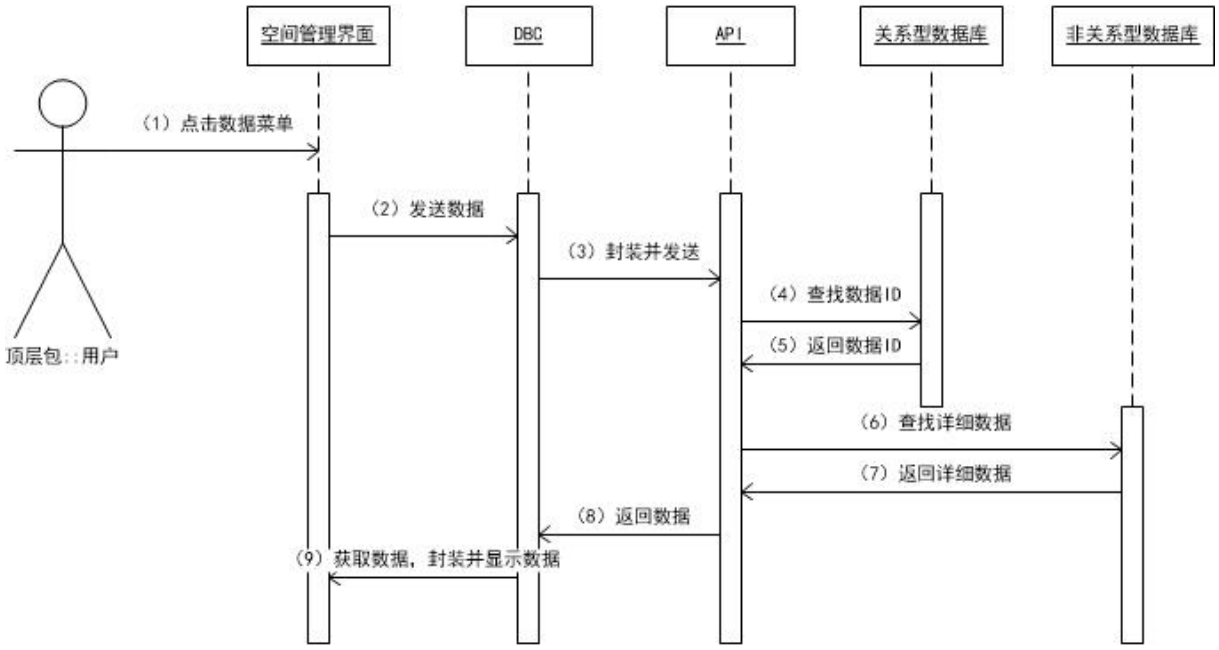


图 4-2 空间管理模块系统顺序图

系统的顺序如下：

用户点击数据菜单，首先查看用户是否登录，若已登录，则进入到我的数据页面；若未登录，则进入到登录界面，登陆之后即进入我的数据页面。进入该页面之后，前端将用户的信息和当前选择的数据类型（全部、共享给我的等）发送到后台，后台的 DBC 模块获取到前端的请求，将信息发送到 API 模块，API 模块根据用户 ID 和操作类型，从关系型数据库中读取所需数据的 ID，再根据 ID 从非关系型数据库中读取数据的详细信息，将信息发送到 DBC 模块，最后将信息发送到前端。前端在获取到信息之后，根据信息的类型，封装到不同的卡片中，并根据时间倒序显示信息。用户在通过目录选择其他类型的数据后，重复上述操作，即可显示所选择的数据。

4.1.4 类图

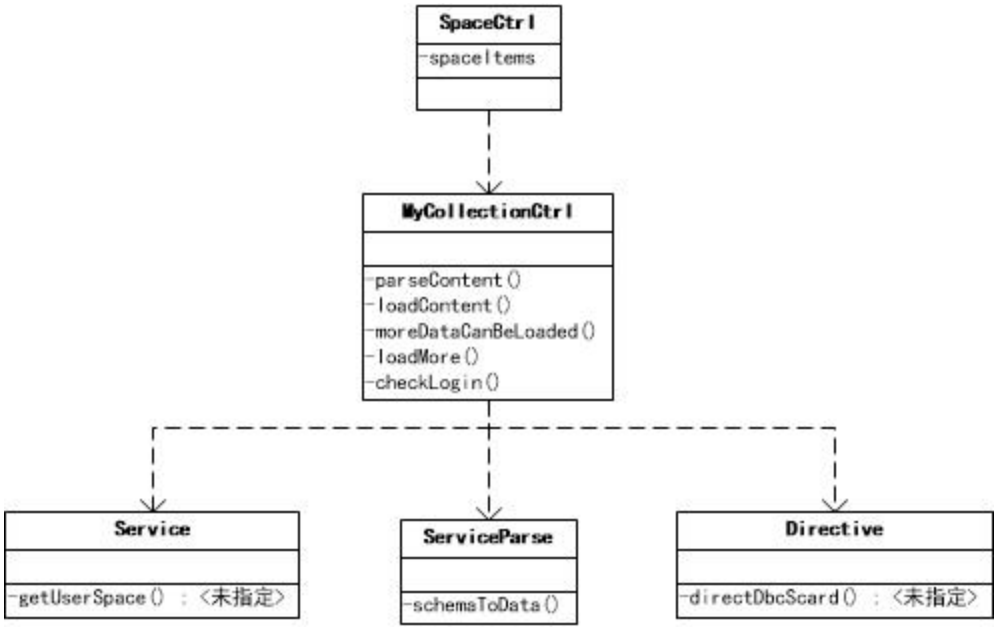


图 4-3 空间管理模块类图

SpaceCtrl 为空间管理模块的总控制类，主要功能为控制 MycollectionCtrl 类以及存储数据模型。

MycollectionCtrl 类为数据控制类，包括数据的后台读取、解析和封装。

Service 类为网络服务类，主要功能为网络连接与传输，控制数据的获取。

ServiceParse 类为数据处理类，主要控制数据的解析。

Directive 类为数据分发类，主要控制数据的封装，并将数据发送到界面。

（注： Service 类、ServiceParse 类、Directive 类在所有模块中通用，故每个模块只介绍该类在当前模块中用到的方法。其他模块同理，不作标注。）

4.1.5 实现效果图

图 4-4 空间管理模块-数据列表界面图

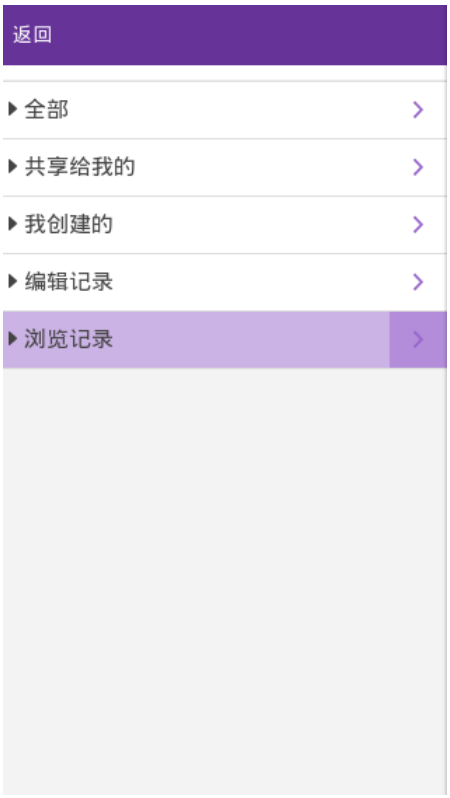


图 4-5 空间管理模块-数据类型列表界面图

4.2 表单管理模块

4.2.1 设计描述

表单管理模块的设计分为前端和后台两个方面。

首先是前端设计。前端主要负责表单的编辑功能，提供了一个创建表单的界面，用户可以通过点击新增字段按钮增加新的字段，也可以通过点击每个字段的编辑按钮对每个字段的属性、限制条件进行编辑。

然后是后台设计。后台主要负责表单的存储，用户在前端创建、编辑完成表单之后，将表单的格式数据发送到后台，并将表单 ID 和用户 ID 存储在关系型数据库的二维关系表中，最后将具体的表单格式数据存储在非关系型数据库中。与空间管理模块类似，后台也分为 DBC、API、存储等模块。

4.2.2 模块流程图

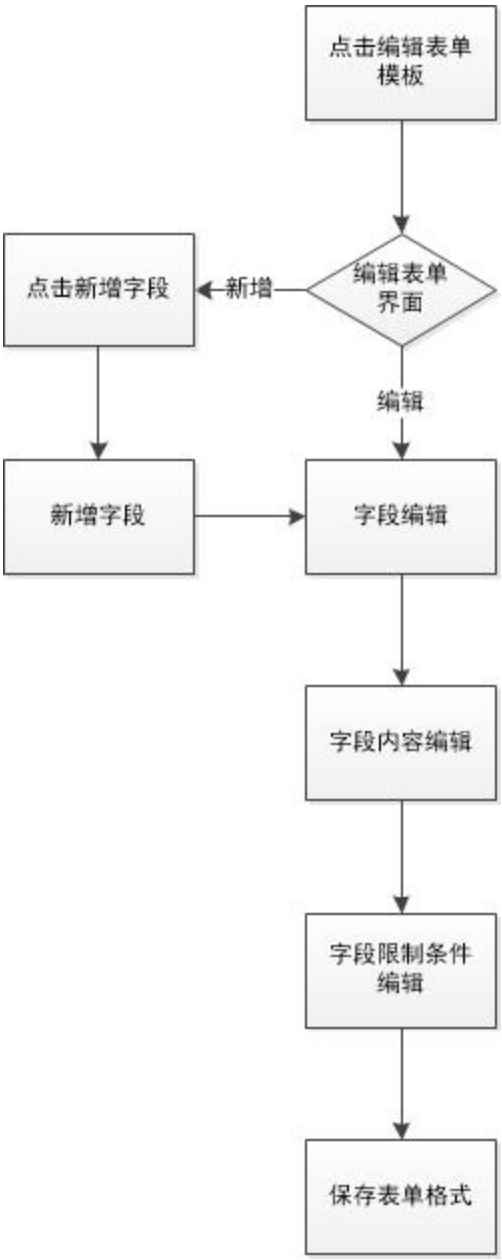


图 4-6 表单管理模块流程图

系统的流程如下：

用户点击编辑表单模板，进入编辑表单界面。此时用户可以新建表单中的字段，也可以对已有的字段进行编辑。点击新增字段后，用户可以新增一条字段，并设置字段类型。点击字段编辑后，用户可以对字段的基本属性（名称、说明、描述）以及限制条件。编辑完成后用户可以点击保存按钮，将表单格式保存到后台。（有关字段类型和限制条

件，详见本模块需求分析）

4.2.3 模块系统顺序图

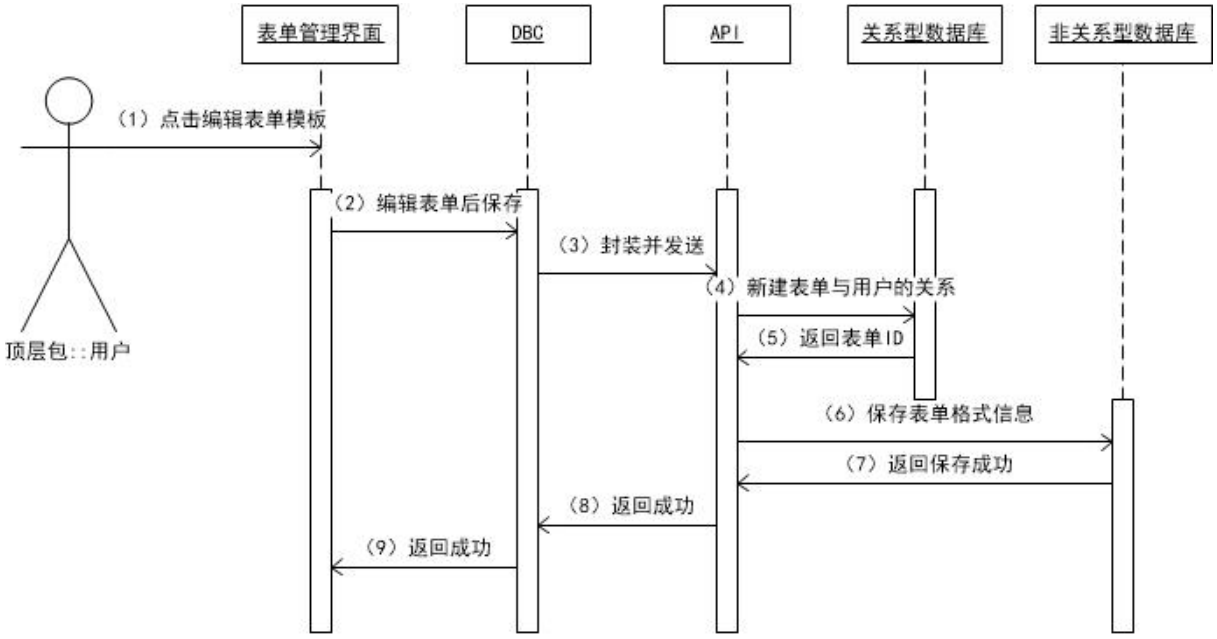


图 4-7 表单管理模块系统顺序图

系统的顺序如下：

用户点击编辑表单模板，进入编辑表单界面，在新增字段、编辑字段限制条件之后，点击保存按钮，前端首先会对字段的限制条件进行字段验证，判断字段的属性是否合法，然后会将表单的格式数据与用户信息发送到 DBC 模块，DBC 模块对数据处理后发送到 API 模块，API 模块会将用户 ID 保存到二维关系表中，并在其中新建表单的 ID，保存完成后将 ID 返回到 API 模块，API 模块将表单的 ID 和格式数据存储在非关系型数据库中，最后后端会将保存成功的信息发送到前端，对用户进行提示。

4.2.4 类图

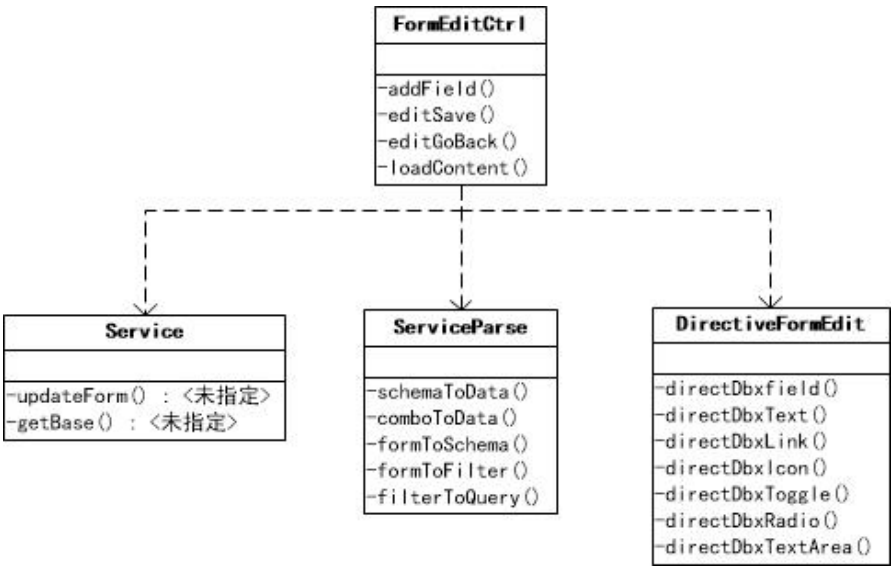


图 4-8 表单管理模块类图

FormEditCtrl 类为表单管理类，管理表单的编辑、字段的新增与编辑和表单的存储。

Service 类为网络服务类，主要控制数据的读取和存储。

ServiceParse 类为数据处理类，主要解析获取到数据。

DirectiveFormEdit 类为数据分发类，主要将解析后的数据根据所需的类型封装、发送到界面。

4.2.5 实现效果图

图 4-9 表单编辑模块-表单信息编辑界面图

图 4-10 表单编辑模块-字段设置界面图

4.3 权限管理模块

4.3.1 设计描述

权限管理模块结构比较复杂。首先，权限是依附于每一个资源的，即直接依附于库、目录和条目，权限从低到高，分别为：不可见、摘要可见、详情可见、创建、编辑、编辑审核、管理员、所有者。关于每个权限的详情已在需求分析中提出，不作赘述。

其次，在详细设计时，首先应遵循以下几个原则：

1. 默认上级权限原则

若新建库、条目或目录时没有具体分配权限给项目，那么新建的库、条 目或目录的权限按照以下规则继承：

（1） 条目继承所属库的权限。（上级权限修改后，继承上级权限的内容权限也对应自动修改）

（2） 目录继承所属库的权限。

（3） 库继承所属空间的权限。（上级权限修改后继承上级权限的内容权限不会自动修改）

权限示例：

用户 “王五” 对库 “技术部库” 拥有 “编辑审核” 权限；

新建 “技术部库” 下条目 “条目 1”；

用户 “王五” 对 “条目 1” 默认拥有编辑审核权限。

2. 细度优化原则

若重新给库或条目分配权限，以重新分配的权限为准。例如，用户在库中拥有低级权限，但在条目中拥有高级权限，则对条目的权限为高级权限，不受库权限影响。

权限示例：

用户 “王五” 对库 “技术部库” 拥有 “编辑审核” 权限；

设置 用户 “王五” 在 “技术部库” 下条目 “条目 1” 权限为 “详情浏览”；

用户 “王五” 对 “条目 1” 拥有 “详情浏览” 权限，无 “编辑审核” 权限。

3. 存在开关原则

开关 1：是否可以重新定义库下条目、目录的权限

权限示例:

用户 “王五” 对 “技术部库” 拥有管理员权限，并设置 “开关 1” 为是；

用户 “李四” 在 “技术部库” 下” 条目 “条目 1”;
用户 “李四” 可以更改 “条目 1“ 的权限。

开关 2 ： 库管理员是否可以看见库下无权限的条目
权限示例:
用户 “王五” 对 “技术部库” 拥有管理员权限，并设置 “开关 2”为否;
用户 “李四” 在 “技术部库” 下创建条目 “条目 1”;
用户 “李四” 更改 “条目 1” 的 “全文浏览” 权限为仅限 “李四” 本人;
用户 “王五” 对 “条目 1” 无全文浏览权限。

4. 存在自动处理器原则

类似批处理插件，可以处理一批下级的权限，选择器可以是库，也可以是目录。批处理效果为一次性作用，下级的权限之后还可以单独更改，只能刷有权限的目录或条目。

权限示例：
用户 “王五” 对 “技术部库” 拥有管理员权限，设置 “技术部库”的 “全文浏览” 权限为 公开;
用户 “王五” 在 “技术部库” 创建 目录 “目录 1”;
用户 “王五” 在 “技术部库” 目录 “目录 1” 下创建条目 “条目 1”, “条目 2”, “条目 3”;
用户 “王五” 使用批处理插件 修改 “目录 1” 下的所有条目的 “全文浏览” 权限为 “注册用户”。
“条目 1”, “条目 2”, “条目 3” 的 “全文浏览” 权限为 “注册用户”。

最后，权限管理也分为前端和后台两个部分。

前端的主要功能是提供了一个对一个资源进行权限编辑的界面，如果用户拥有编辑权限的权限，则可以对本资源的权限进行修改。

后台的主要功能分为两类，一类是与其他模块的配合，在其他模块对资源进行操作的时候进行权限验证，每一个操作都需要验证通过后才可以继续执行；另一类是配合前端的权限编辑，对权限进行修改和存储。

4.3.2 模块流程图

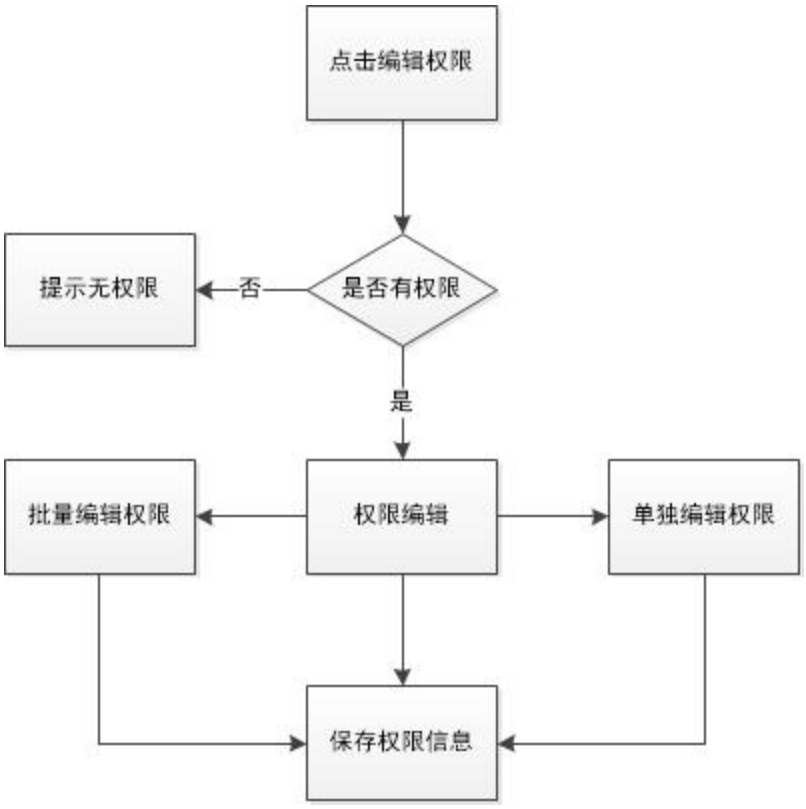


图 4-11 权限管理模块流程

系统的流程如下：

用户在点击编辑权限后，如果有权限则会进入权限编辑页面。在该页面中，可以对权限信息进行编辑。对于摘要可见和全文浏览权限，可以将其设置成全网公开、全部注册用户可见和仅限以下用户可见三种状态；对于编辑权限、管理员权限和所有者权限，可以将其设置为仅限以下用户可操作。在编辑完成后，点击保存按钮，权限信息将发送到后台并保存。

4.3.3 模块系统顺序图

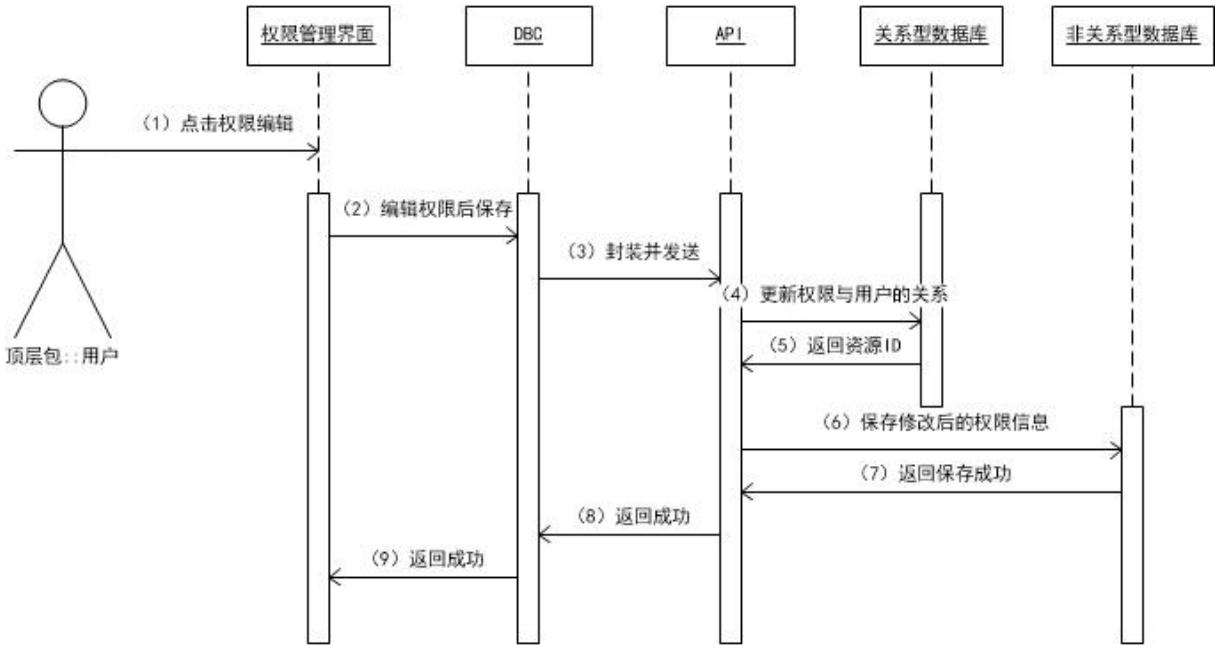


图 4-12 权限管理模块系统顺序图

系统的顺序如下：

用户点击权限编辑按钮，进入到权限管理界面，对权限进行编辑以后，前端会将用户数据和权限数据发送到 DBC 模块，DBC 对用户 ID 进行验证，确认具有编辑权限的权限后，会将用户数据和权限数据封装并发送到 API 模块，API 模块会在关系型数据库中更新权限与用户的关系，并返回权限所属资源的 ID，然后 API 模块会将修改后的权限信息保存到非关系型数据库中，并返回保存成功的信息，最后 DBC 模块会将信息发送到前端。

4.3.4 类图

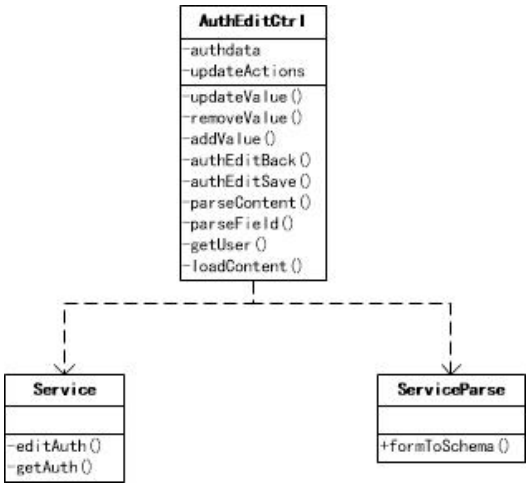


图 4-13 权限管理模块类图

AuthEditCtrl 类为权限管理类，管理权限的编辑。

Service 类为网络服务类，主要控制数据的读取和存储。

ServiceParse 类为数据处理类，主要解析获取到数据。

4.3.5 实现效果图

图 4-14 权限管理模块-权限编辑界面图

4.4 社区管理模块

4.4.1 设计描述

社区管理模块包括评论、点赞、通知、论坛四个功能。

评论功能：用户在资源详情页面下，可通过点击评论按钮，对资源进行评价，评价完毕后点击保存按钮，评论的数据将会发送到后台并保存。在评论的过程中，用户可以提到其他用户，即@的功能。

点赞功能：用户可以对资源进行点赞，点赞信息将会保存到二维关系表中。

通知功能：用户可以收到别人发送的消息，也会收到系统的通知。通知都会被保存到后台。

论坛：用户可以在系统中自由创建话题，话题分为两类，一类是基于库、条目的话题，这类话题会有和库、条目的链接，一般用于讨论与资源相关的问题，每一个；另一类是不基于任何内容的话题。每一个话题都类似于一个贴子，可以在其中自由评价、点赞。每一个话题都会被存储到后台非关系型数据库中。

4.4.2 模块流程图

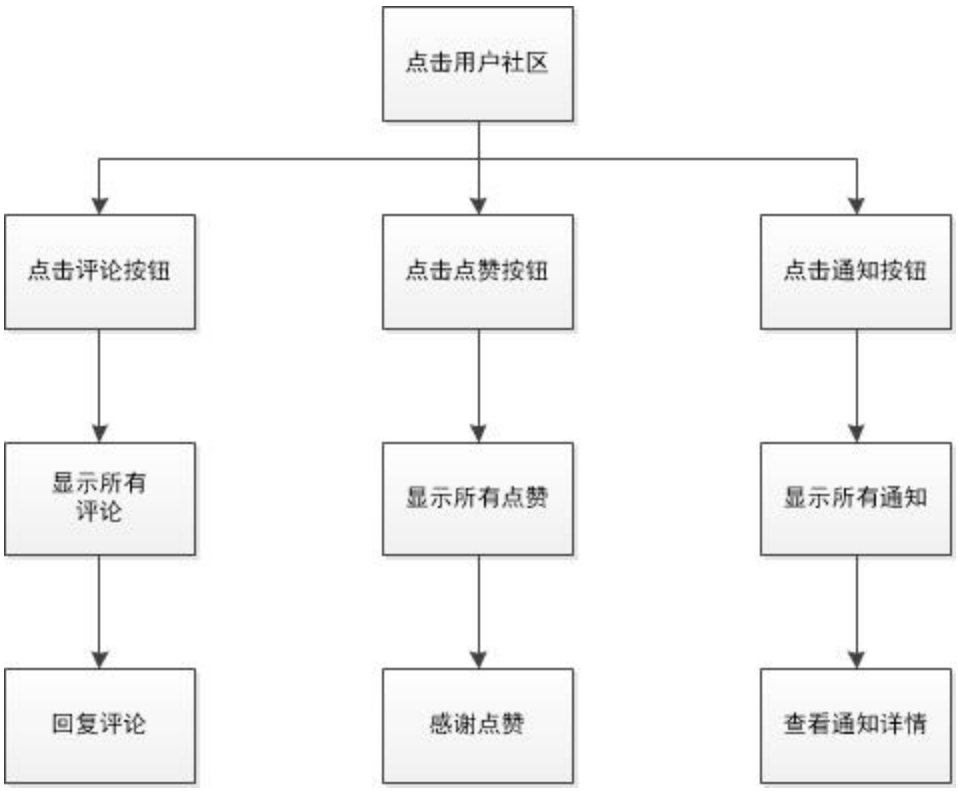


图 4-15 社区管理模块社区菜单流程图

该流程为用户社区菜单的流程。用户点击社区按钮后，进入社区菜单界面，界面提供评论、点赞和通知三个按钮，并在按钮下方显示其他用户发送的信息。 用户点击评论按钮后，会根据时间倒序显示其他用户对我创建的资源的评论，以及所有提到我的评论。点击其中一条评论后，用户可以对其进行回复。

用户点击点赞按钮后，会根据时间倒序显示其他用户对我创建的资源的点赞记录。点击其中一条点赞后，用户可以对点赞的用户进行感谢。

用户点击通知按钮后，会根据时间倒序显示所有系统通知。点击其中一条通知后，可以查看该通知的详情。

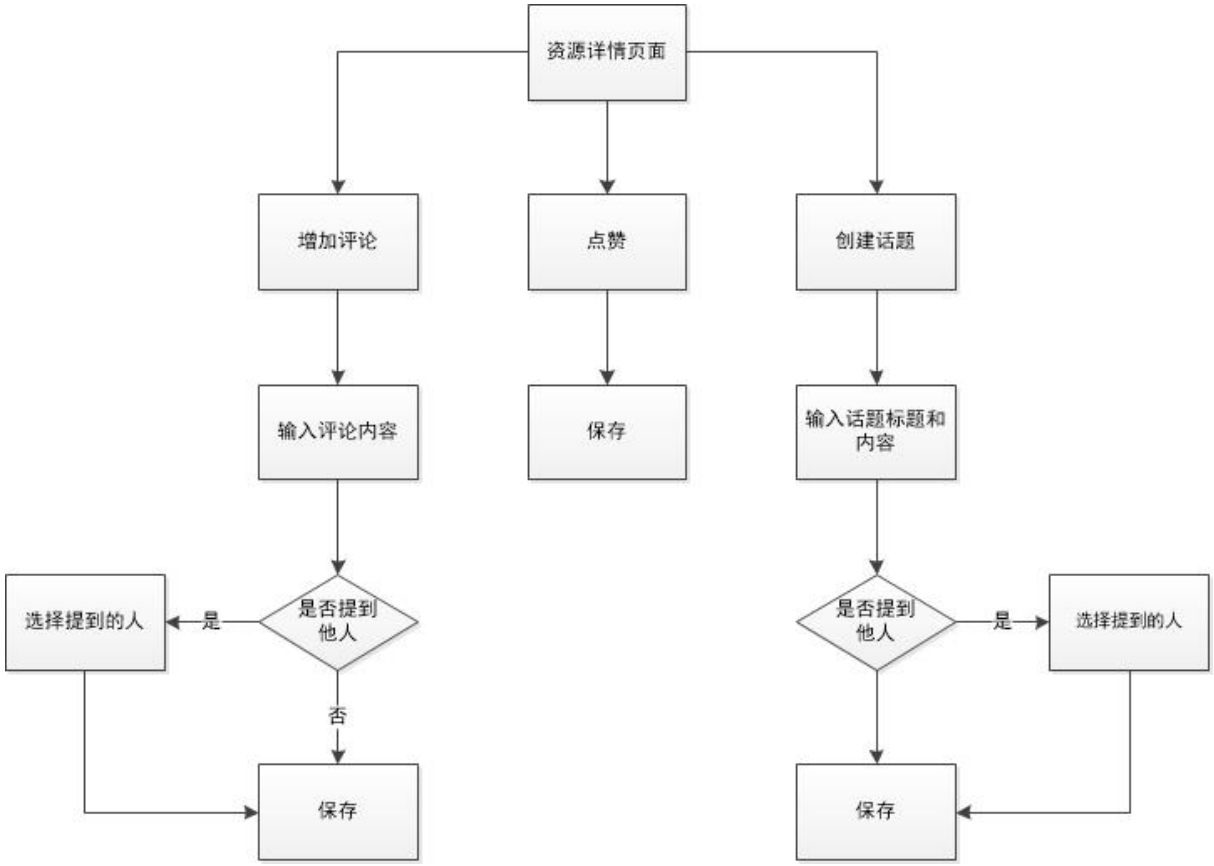


图 4-16 社区管理模块资源详情流程图

该流程为资源详情界面下的评论、点赞和创建话题的流程。

用户点击评论按钮后，可以对当前资源进行评论，也可以提到他人（@），系统会将评论内容保存，并提示资源的创建者和提到的人。

用户点击点赞按钮后，可以对当前资源进行点赞，系统会将点赞信息保存，并提示资源的创建者。

用户点击创建话题按钮后，可以创建一个基于当前资源的话题，并写下第一条内容，也可以提到他人，系统会将话题保存，并提示提到的人。

4.4.3 模块系统顺序图

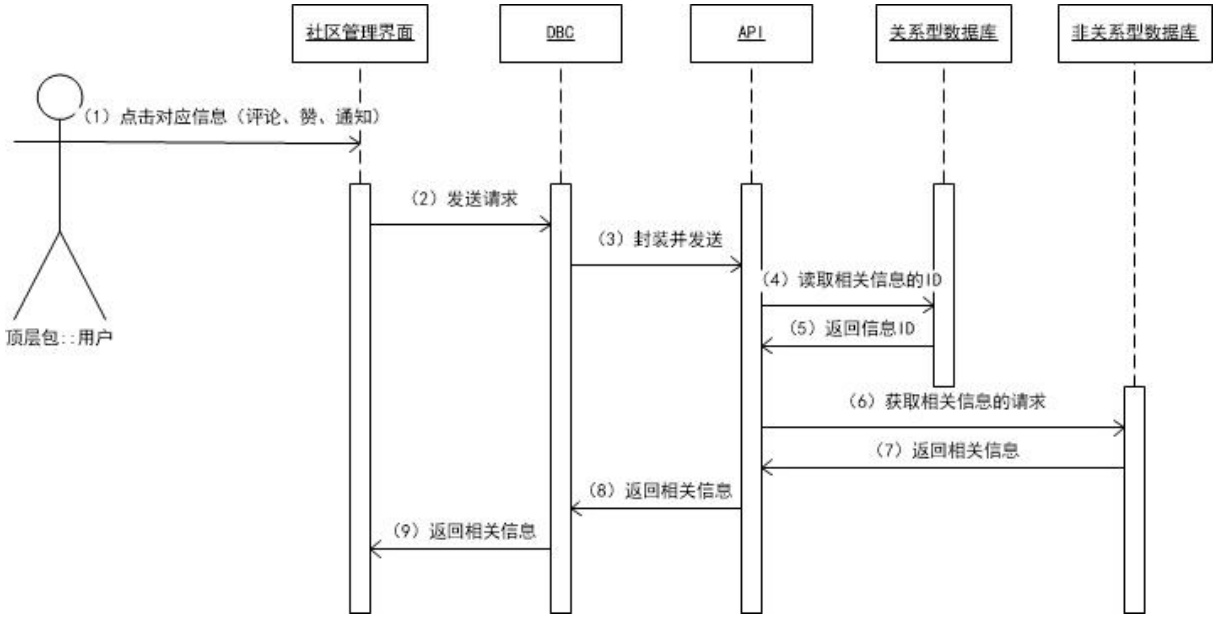


图 4-17 社区管理模块社区菜单系统顺序图

该顺序为用户社区菜单的顺序。用户在社区菜单界面下，点击评论、赞、通知三个按钮，可分别发送获取对应信息的请求到后台，后台在关系型数据库中获取到资源的 ID 后，会从非关系型数据库中获取信息的详细内容，并返回到前端。

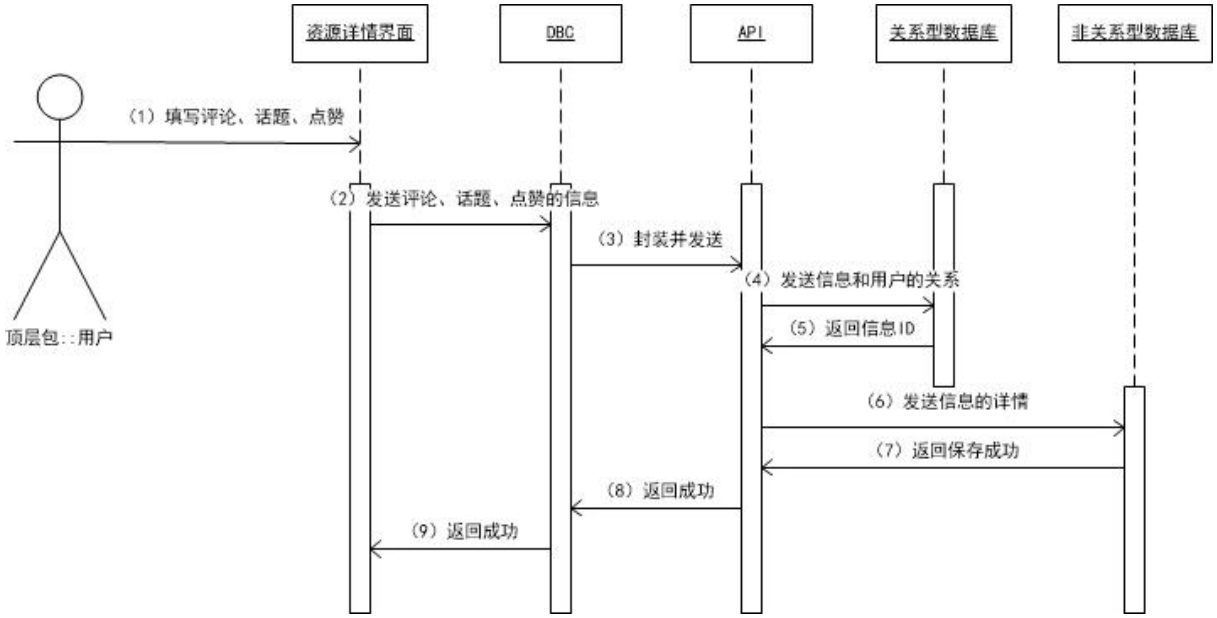


图 4-18 社区管理模块资源详情系统顺序图

该顺序为资源详情界面下的评论、点赞和创建话题的顺序。用户在该界面下，点击评论按钮可以输入评论信息，然后点击保存按钮，信息将被发送到 DBC 模块，在模块中进行格式化处理后，发送到 API 模块，并将信息与用户（创建者和提到的人）的关系存储到关系型数据库中，然后将信息存储到非关系型数据库中。最后，后台将会发送提示信息给提到的用户以及被评论的用户。

4.4.4 类图

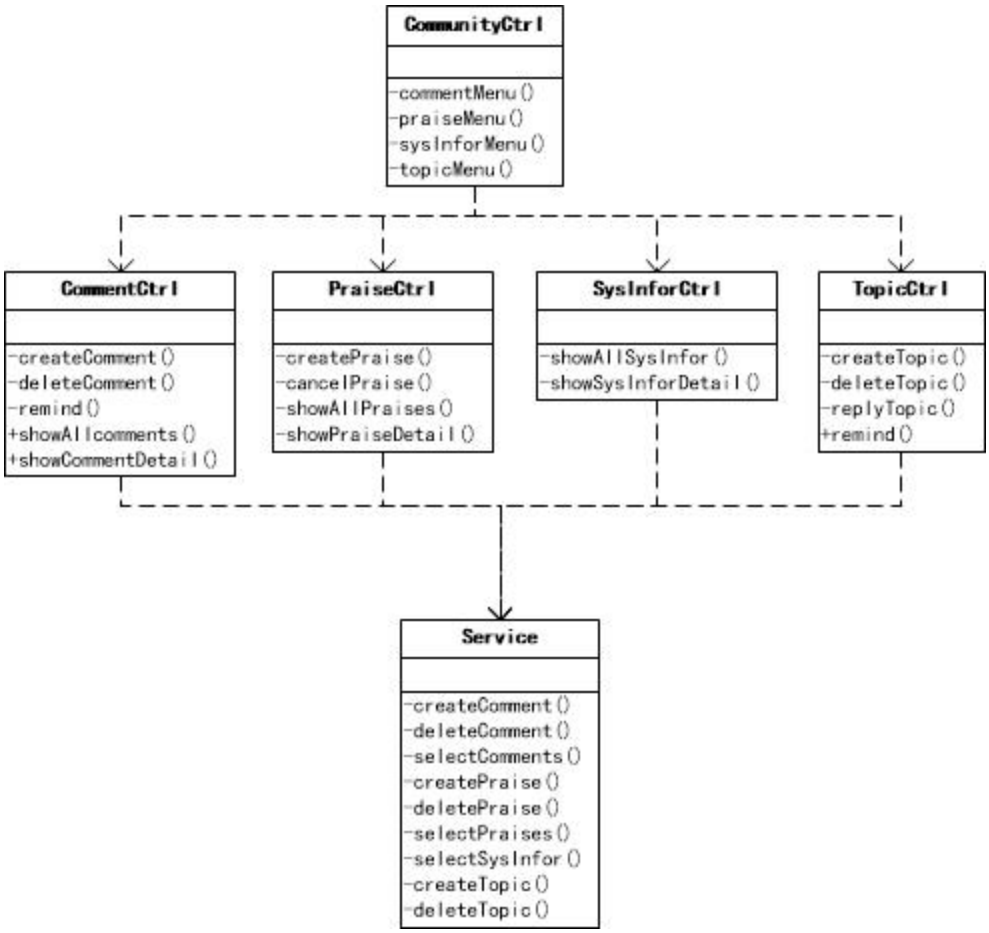


图 4-19 社区管理模块类图

CommunityCtrl 类为社区菜单控制类，主要功能为控制社区的四个功能调用。

CommentCtrl 类为评论控制类，主要功能为控制评论的新建、删除、提到功能和显示评论信息。

PraiseCtrl 类为点赞控制类，主要功能为点赞、取消点赞、显示点赞。

SysInforCtrl 类为系统信息控制类，主要功能为显示系统信息。

TopicCtrl 类为话题控制类，主要功能为创建、删除、回复话题以及提到用户的功能。

Service 类为网络服务类，主要处理数据的发送和接收。

4.4.5 实现效果图

图 4-20 社区管理模块-社区管理菜单界面

图 4-21 社区管理模块-点赞信息列表界面

5 总结

本文以“数据酷”网络数据管理平台的设计与实现为基础撰写。刚开始着手这个项目时，项目组首先分析了当今时代背景下，数据管理的整体趋势以及存在的问题，并调研了现有的几款网络数据管理平台的优缺点，提出了本项目的需求。然后，根据软件实际使用中，需要面对的一些技术上的问题，对系统的整体架构进行分析，解决的主要问题是在数据存储方面，决定采用关系型和非关系型数据库结合的方式，使操作更加自由化。最后到具体实现部分，将系统分解成了八个模块，每个模块中又分别包含了 UI 层、DBC 层、API 层、存储层四个层次，将功能进行了封装，使得功能更加独立，便于开发。

具体到论文上，本人从背景、需求分析、系统架构、模块设计与实现几个方面来对本项目进行阐述。

首先是背景，主要从时代背景、研究现状、研究意义几个方面对现状进行了简单的介绍，说明了开发本项目的内容和意义，使读者对本项目有一个整体的了解，便于理解该论文的内容。

然后是需求分析。需求分析从前面分析的背景出发，先分析了现有数据管理平台的主要功能以及存在的问题，然后根据存在的问题，提出新的解决方案。在完善了功能方面的需求之后，论文又根据该需求对技术上的一些内容作了定义，在技术层面进行了整体的需求简述。接着是对系统四个模块的需求描述，采用表格、用例图、用例描述等多种形式对需求进行阐述。

接着是系统架构。系统架构先说明了系统的整体架构，使用架构图描述了系统采用的 B/S 架构。然后说明了系统四个层次各自的架构，其中重点说明的是数据存储层的架构，使用表格的形式解释了关系型和非关系型数据库结合存储的方式，方便读者后面的详细设计。接下来介绍了系统使用的关键技术和开发环境。

最后是模块设计与实现。模块设计与实现采用设计描述、流程图、系统顺序图、类图等形式对四个模块进行了描述，介绍了每个模块的详细实现过程。

在项目研发和撰写论文的过程中，我学到了很多。在技术方面，学会了 ionic、Spring boot、NoSQL 等过去没有接触过的技术，并在团队开发的过程中，巩固并实践了过去自己所了解的软件工程开发的流程。在其他方面，学会了团队合作、沟通的技巧。同时，我也认识到自己的许多不足之处，希望在未来的工作与学习中，逐渐改正这些问题，让自己一点点地进步。

参考文献

[1]《计算机安全》编辑部. 迎接大数据时代[J]. 计算机安全, 2013, 4(1): 1-2

[2]Jeremy, Wilken. Ionic in Action: Hybrid Mobile Apps with Ionic and Angularjs[M]. USA:Manning Publications, 2015.

[3]Greg, L, Turnquist. Learning Spring Boot[M]. USA:Packt Publishing - ebooks Account, 2014.

[4]Adam, Freeman. 张桐. 张铮铮. AngularJS 高级程序设计[M]. 北京:人民邮电出版社, 2015.

[5]鸟哥. 鸟哥的Linux私房菜:基础学习篇(第3版)[M]. 北京:人民邮电出版社, 2010.

[6]Bernard, Golden. Amazon Web Services For Dummies[M]. USA:For Dummies, 2013.

[7]Leonard, Richardson. Mike, Amundsen. 赵震一. 李哲. RESTful Web APIs 中文版[M]. 北京:电子工业出版社, 2014.

[8]汪云飞. JavaEE开发的颠覆者: Spring Boot 实战[M]. 北京:电子工业出版社, 2016.

[9]李子骅. Redis 入门指南（第 2 版）[M]. 北京:人民邮电出版社, 2015.

[10] Ezra, Schwartz. Axure RP 6 Prototyping Essentials[M]. USA:Packt Publishing, 2012.

[11]高利雅. 基于 solr 全文搜索引擎的研究与实现[D]. 成都:电子科技大学, 2014.

[12]王瑛. 一种基于 Redis 的消息服务模块的设计与实现[D]. 北京:北京大学, 2013.

[13]苏婵. 非关系型数据库及在文档库管理平台中的研究与应用[D]. 南昌:南昌大学, 2014.

[14]刘一梦. 基于 MongoDB 的云数据管理技术的研究与应用[D]. 北京:北京交通大学, 2012.

致　谢

在论文撰写的过程中，要感谢许多的人。作为一个初学者，这个过程是很艰难，如果没有所有老师、同事、同学的帮助，很多问题都不能解决。正是你们的帮助，让我解决了问题，最终顺利完成论文。

首先，我要感谢我的指导老师！谢谢老师的悉心教导。在整个毕业设计的过程中，我的导师对我论文中的问题一直很上心，很用心地帮我想解决问题的办法。每一次去找导师沟通，导师都很耐心地解答我的问题，对此我深怀感激！

接下来要感谢的是我的同事、同学。在项目过程中，我的同事都很热心地帮我解决项目中的问题，我的同学也为论文的撰写提供了宝贵的意见！

最后，感谢北京交通大学所有指导我、帮助我的人，正是你们无私地帮助，让我顺利地在这里成长！

附 录

附录 A：AngularJS：JavaScript 中的现代 MVC 框架

AngularJS: A Modern MVC Framework in JavaScript

Abstract: AngularJS is a JavaScript MVC Framework created by Google to build properly architecture and maintainable web application. AngularJS is built around the philosophy that declarative code is better than imperative code while building UIs and wirin g different components of web applications together. In this article we have shown the features of AngularJS.

INTRODUCTION

AngularJS is not a library rather AngularJS is a JavaScript framework that embraces extending HTML into a more expressive and readable format. It allows you to decorate your HTML with special markup that synchronizes with your JavaScript leaving you to write your application logic instead of manually updating views. Whether you're looking to augment existing JavaScript applications or harness the full power of the framework to create rich and interactive SPA's, Angular can help you write cleaner and more efficient code. This one may seem obvious, but it's important to remember that many (not all) frameworks are made by hobbyists in the open source community. While passion and drive have forged frameworks, like Cappucino and Knockout, Angular is built and maintained by dedicated (and highly talented) Google engineers. This means you not only have a large open community to learn from, but you also have skilled, highly-available engineers tasked to help you get your Angular questions answered.

This isn't Google's first attempt at a JavaScript framework; they first developed their comprehensive Web Toolkit, which compiles Java down to JavaScript, and was used by the Google Wave team extensively. With the rise of HTML5, CSS3, and JavaScript, as both a front-end and back-end language, Google realized that the web was not meant to be written purely in Java.

Why Choose AngularJS?

1. DOM has markup [Angular markup lives in the DOM]
2. Data is POJO [Angular uses plain old JavaScript objects]
3. DI for modules [Angular heavily leverages Dependency Injection]

1. DOM has markup:

```
<!DOCTYPE html>
<html>
<head>
    <script src="/js/angular.js"></script>
    <script>
        var app = angular.module('MyApp', []);
    </script>
</head>
<body ng-app="MyApp">

    <input type="text" ng-model="someval" />

    <my-custom-tag>{{ someval }}</my-custom-tag>

</body>
</html>
```

Templates in most client-side JavaScript frameworks work like something like this:

template with markup→framework template engine→HTML→DOM

Angular, on the other hand puts markup directly into the HTML document and the flow looks like this:

HTML with Angular markup→DOM→Angular template engine
Angular evaluates the markup only after HTML has been loaded into the DOM.
This approach has three major benefits.

1.Integration with Existing Apps.Since Angular only starts evaluating the page at the end of the loading process (i.e. once HTML is in the DOM), it is very easy to sprinkle small bits of Angular "magic" on top of existing applications.
2.Simplicity.You can work with Angular in a basic HTML document from you local file system. Just open the HTML document in your browser. No need for any web server or template build process. I have found this very useful for creating quick mockups of a new website or piece of functionality.
3.Extensibility.Using Directives, Angular
allows you to create custom elements and attributes that extend the standard HTML vocabulary. For example, in this slide there is a my-custom-tag element. Using Angular you can define how that element is rendered and assign behaviors to it. This allows you to create a library of your own reusable components.

2.Data is POJO:

```
    <script>
        var app = angular.module('MyApp', []);
        app.controller('MyCntl', function ($scope) {
            $scope.guys = ['jeff', 'joe', 'bob'];
        })
    </script>
</head>
<body ng-app="MyApp">
    <ul ng-controller="MyCntl">
        <li ng-repeat="guy in guys">{{ guy }}</li>
    </ul>
</body>
```

Angular is one of the only major front end frameworks that utilize plain old Javascript objects (POJOs) for the model layer. This makes it extremely easy to integrate with existing data sources and play with basic data.
Let's say you make an AJAX call to get some data from an API. Before you can bind that data to the DOM, most frameworks require you to wrap the data in Model objects that have getters and setters. The getters/setters are how non-Angular frameworks propagate data change events.
Angular gets around this by using a process called dirty checking where snapshots of data over time are compared to see if anything has changed. While there are certainly some downsides to this approach (ex. $scope.$apply, data binding limits, etc.)

3. DI for modules:

```
var app = angular.module('MyApp', []);

app.factory('changeUrlTo', function ($location) {
    return function (destinationUrl) {
        $location.path(destinationUrl);
    };
});

app.controller('MyCntl', function (changeUrlTo) {
    changeUrlTo('/another-page');
});
```

There are some people that love dependency injection and there are some people that hate it. If you are going to work with Angular, you sort of need to be in the former camp. I personally love it because it promotes better modularization of code and enables strong unit testing.
Unit testing front end code is usually hard because there are so many sticky dependencies. Angular's DI allows you to mock out many of these dependencies and isolate individual components.

FEATURES OF ANGULARJS

FEATURE 1: TWO WAY DATA-BINDING
Think of your model as the single-source-of-truth for your application. Your model is where you go to to read or update anything in your application.
Data-binding is probably the coolest and most useful feature in AngularJS. It will save you from writing a considerable amount of boilerplate code. A typical web application may contain up to 80% of its code base, dedicated to traversing, manipulating, and listening to the DOM. Data- binding makes this code disappear, so you can focus on your application.
Think of your model as the single-source-of-truth for your application. Your model is where you go to to read or update anything in your application. The data-binding directives provide a projection of your model to the application view. This projection is seamless, and occurs without any effort from you.
Traditionally, when the model changes, the developer are responsible for manually manipulating the DOM elements and attributes to reflect these changes. This is a two-way street. In one direction, the model changes drive change in DOM elements. In the other, DOM element changes necessitate changes in the model. This is further complicated by user interaction, since the developer is then responsible for interpreting the interactions, merging them into a model, and updating the view. This is a very manual and cumbersome process, which becomes difficult to control, as an application grows in size and complexity.
There must be a better way! AngularJS' two-way data-binding handles the synchronization between the DOM and the model, and vice versa.
Here is a simple example, which demonstrates how to bind an input value to an <h1> element.

```
<!doctype html>
<html ng-app>
    <head>
        <script src=”http://code.angularjs.org/angular-          1.0.0rc10.min.js”>
        </script>
    </head>
    <body>
        <div>
            <label>Name:</label>
            <input type=”text” ng-model=”yourName” placehold=”Enter a name here”>
            <hr>
            <h1>Hello,{{yourName}}!</h1>
        </div>
    </body>
</html>
```

This is extremely simple to set up, and almost magical…

FEATURE 2: TEMPLATES
It's important to realize that at no point does AngularJS manipulate the template as strings. It's all the browser DOM.
In AngularJS, a template is just plain-old-HTML.The HTML vocabulary is extended, to contain instructions on how the model should be projected into the view.
The HTML templates are parsed by the browser into the DOM. The DOM then becomes the input to the AngularJS compiler. AngularJS traverses the DOM template for rendering instructions, which are called directives. Collectively, the directives are responsible for setting up the data- binding for your application view.

It is important to realize that at no point does AngularJS manipulate the template as strings. The input to AngularJS is browser DOM and not an HTML string. The data-bindings are DOM transformations, not string concatenations or innerHTML changes. Using the DOM as the input, rather than strings, is the biggest differentiation AngularJS has from its sibling frameworks. Using the DOM is what allows you to extend the directive vocabulary and build your own directives, or even abstract them into reusable components!
One of the greatest advantages to this approach is that it creates a tight workflow between designers and developers. Designers can mark up their HTML as they normally would, and then developers take the baton and hook in functionality, via bindings with very little effort.
Here is an example where I am using the ng-repeat directive to loop over the images array and populate what is essentially an img template.

```
function AlbumCtrl($scope) {
    scope.images = [
        {"thumbnail":"img/image_01.png",    "description":"Image 01 description"},
        {"thumbnail":"img/image_02.png",    "description":"Image 02 description"},
        {"thumbnail":"img/image_03.png",    "description":"Image 03 description"},
        {"thumbnail":"img/image_04.png",    "description":"Image 04 description"},
        {"thumbnail":"img/image_05.png",    "description":"Image 05 description"}
    ];
}
<div ng-controller="AlbumCtrl">
    <ul>
        <li ng-repeat="image in images">
            <img ng-src="{{image.thumbnail}}" alt="{{image.description}}">
        </li>
    </ul>
</div>
```

It is also worth mentioning, as a side note, that AngularJS does not force you to learn a new syntax or extract your templates from your application.

FEATURE 3: MVC
AngularJS incorporates the basic principles behind the original MVC software design pattern into how it builds client-side web applications.
The MVC or Model-View-Controller pattern means a lot of different things to different people. AngularJS does not implement MVC in the traditional sense, but rather something closer to MVVM (Model-View-ViewModel).

The Model
The model is simply the data in the application. The model is just plain old JavaScript objects. There is no need to inherit from framework classes, wrap it in proxy objects, or use special getter/setter methods to access it. The fact that we are dealing with vanilla JavaScript is a really nice feature, which cuts down on the application boilerplate.

The ViewModel
A viewmodel is an object that provides specific data and methods to maintain specific views.
The viewmodel is the $scope object that lives within the AngularJS application. $scope is just a simple JavaScript object with a small API designed to detect and broadcast changes to its state.

The Controller
The controller is responsible for setting initial state and augmenting $scope with methods to control behavior. It is worth noting that the controller does not store state and does not interact with remote services.

The View
The view is the HTML that exists after AngularJS has parsed and compiled the HTML to include rendered markup and bindings.

This division creates a solid foundation to architect your application. The $scope has a reference to the data, the controller defines behavior, and the view handles the layout and handing off interaction to the controller to respond accordingly.

FEATURE 4: DEPENDENCY INJECTION

AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.
Dependency Injection (DI) allows you to ask for your dependencies, rather than having to go look for them or make them yourself. Think of it as a way of saying "Hey I need X', and the DI is responsible for creating and providing it for you.

```
function EditCtrl($scope， $location， $routeParams) {
    // Something clever here...
}
```

You are also able to define your own custom services and make those available for injection as well.

```
angular.module('MyServiceModule',[]).factory('notify'， ['$window', function (win) {
    return function (msg) {
        win.alert(msg);
    };
}]);
function myController(scope， notifyService) {
    scope.callNotify = function (msg) {
        notifyService(msg);
    };
}
myController.$inject = ['$scope'， 'notify'];
```

FEATURE 5: DIRECTIVES
Directives are my personal favorite feature of AngularJS. Have you ever wished that your browser would do new tricks for you? Well, now it can! This is one of my favorite parts of AngularJS. It is also probably the most challenging aspect of AngularJS.
Directives can be used to create custom HTML tags that serve as new, custom widgets. They can also be used to "decorate" elements with behavior and manipulate DOM attributes in interesting ways.
Here is a simple example of a directive that listens for an event and updates its $scope, accordingly.

```
myModule.directive('myComponent'， function(mySharedService) {
    return {
        restrict: 'E'，
        controller: function($scope， $attrs， mySharedService) {
            $scope.$on('handleBroadcast'， function() {
                $scope.message = 'Directive: ' + mySharedService.message;
            });
        }，
        replace: true，
        template: '<input>'
    };
});
```

Then, you can use this custom directive, like so.

```
<my-component ng-model="message"></my-component>
```

Creating your application as a composition of discrete components makes it incredibly easy to add, update or delete functionality as needed.

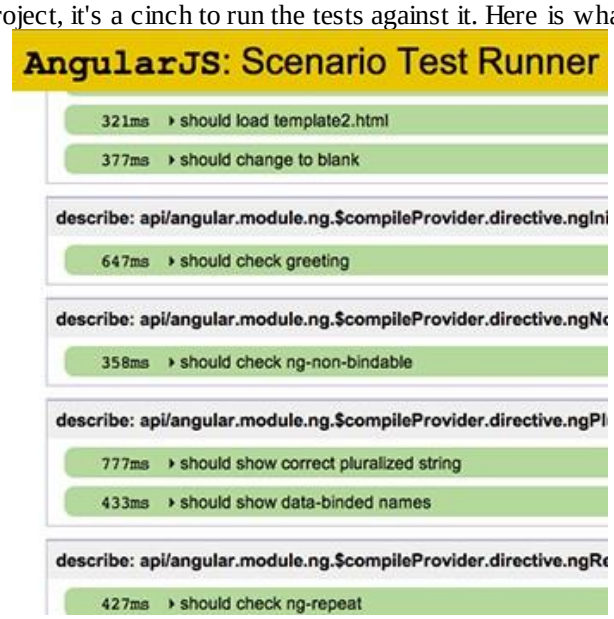
BONUS FEATURE: TESTING

The AngularJS team feels very strongly that any code written in JavaScript needs to come with a strong set of tests. They have designed AngularJS with testability in mind, so that it makes testing your AngularJS applications as easy as possible. So there's no excuse for not doing it.

Given the fact that JavaScript is dynamic and interpreted, rather than compiled, it is extremely important for developers to adopt a disciplined mindset for writing tests.

AngularJS is written entirely from the ground up to be testable. It even comes with an end-to-end and unit test runner setup. If you would like to see this in action, go check out the angular-seed project at https://github.com/angular/angular-seed.

Once you have the seed project, it's a cinch to run the tests against it. Here is what the output looks like:



The API documentation is full of end-to-end tests that do an incredible job of illustrating how a certain part of the framework should work. After a while, I found myself going straight to the tests to see how something worked, and then maybe reading the rest of the documentation to figure something out.

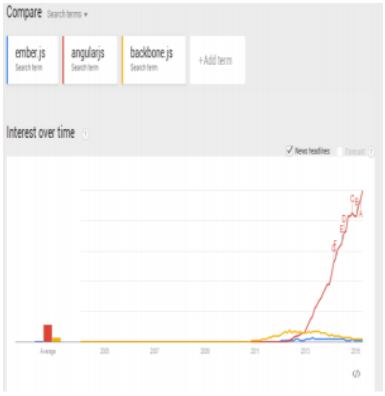
Comparisons in Community:

Community is one of the most important factors to consider when choosing a framework. A large community means more questions answered, more third-party modules, more YouTube tutorials…you get the point. I have put together a table with the numbers, as of August 16, 2014. Angular is definitely the winner here, being the 6th most-starred project on GitHub and having more questions on StackOverflow than Ember and Backbone combined, as you can see below:

Metric	AngularJS	Backbone.js	Ember.js
Stars on Github	27.2k	18.8k	11k
Third-Party Modules	800 ngmodules	236 backplugs	21 emberaddons

StackOverflow Questions	49.5k	15.9k	11.2k
YouTube Results	~75k	~16k	~6k
GitHub Contributors	928	230	393
Chrome Extension Users	150k	7k	38.3k

All those metrics, however, merely show the current state of each framework. It is also interesting to see which framework has a faster-growing popularity. Fortunately, using Google Trends (Till 18/4/2015) we can get an answer for that too:



CONCLUSION:

Angular's innovative approach for extending HTML will make a lot of sense for people who are web developers in soul. With a large community and Google behind it, it is here to stay and grow, and it works well both for quick prototyping projects and large-scale production applications.

REFERENCES:

[1] AngularJS vs. Backbone.js vs. Ember.js (https://www.airpair.com/js/javascript-framework-comparison)
[2] https://www.airpair.com/angularjs/posts/jquery- angularjs-comparison-migration-walkthrough
[3] http://code.tutsplus.com/tutorials/3-reasons-to-choose-a

附录 B：AngularJS：JavaScript 中的现代 MVC 框架——翻译

AngularJS：JavaScript 中的现代 MVC 框架

摘要：AngularJS 是由谷歌创建的，建立可靠架构和维护网页应用程序的 JavaScript MVC 框架。AngularJS 围绕声明式代码比命令式代码更好的理念建立,用于在网页应用程序中建立用户界面和布线不同的组件。在这篇文章中，我们会展示 AngularJS 的特点。

简介

AngularJS 不是一个库，而是延展 HTML，并使其格式更具表现力和可读性的 JavaScript 框架。它允许开发者使用特殊标记来装点自己的 HTML，同步开发者的 JavaScript，让开发者的关注点更多放在应用程序逻辑，而不是手动更新视图。无论你是在尝试改进现有 JavaScript 应用，还是重新开发丰富、可交互的单页网页应用，Angular 都可以帮你编写更加干净、高效的代码。

这似乎是显而易见的，但记住很多（不是全部）框架是由开源社区的编码爱好者开发的。就像卡布奇诺和“敲除”（Corel 公司出品的专业去背景软件）由于激情和动力而被创造，Angular 是由专业而又才华横溢的谷歌工程师打造。这意味着你不仅拥有许多开放的社区来学习它，还可以有熟练，高水平的工程师来负责帮助你解决 Angular 的问题。

这并不是谷歌在 JavaScript 框架中的第一次尝试。他们首先开发了全面的网页工具包，它可以把 Java 编译为 JavaScript，被广泛地应用于 Google Wave 的团队中。随着 HTML5，CSS3 和 JavaScript 的兴起，在前端和后台的开发语言方面，谷歌意识到，网页不必要纯粹使用 Java 编写。

为什么选择 AngularJS 呢？

1. 文档对象模型标记【Angular 标记于文档对象模型中】
2. 数据是简单 Java 对象【Angular 使用简单 Java 对象】
3. 依赖注入模型【Angular 高度切合依赖注入】

1. 文档对象模型标记：

```
<!DOCTYPE html>
<html>
<head>
    <script src="/js/angular.js"></script>
    <script>
        var app = angular.module('MyApp', []);
    </script>
</head>
<body ng-app="MyApp">

    <input type="text" ng-model="someval" />

    <my-custom-tag>{{ someval }}</my-custom-tag>

</body>
</html>
```

客户端 JavaScript 框架的模型都进行如下工作：

标记模型→框架模型引擎→HTML→文档对象模型

另一方面，AngularJS 把构建过程直接放入到 HTML 文本中，流程如下：

Angular 标记 HTML→文档对象模型→Angular 模型引擎

Angular 只在 HTML 将文本对象模型载入之后才进行评定标记，这种方法有 3 个好处。

1. 集成现有的应用程序。由于 Angular 在加载过程结束后开始评估页面（即 HTML 是在 DOM 中一次装载），这很容易在现有应用程序之上添加 Angular“魔法”。
2. 简单。你可以从您的本地文件系统用 Angular 基本的 HTML 文档工作。就在你的浏览器中打开 HTML 文档，无需任何 Web 服务器或模板构建过程。我发现这对于创建一个新的网站或一项功能的快速原型是非常有用的。
3. 可扩展性。使用指令， Angular 允许您创建自定义的元素和属性扩展标准 HTML 词汇。例如，在这个页面中的 my-custom-tag 元素。使用 Angular 可以定义元素的呈现和分配行为。这使你可以创建自己的可重用组件库。

2. 数据是简单 Java 对象：

```
    <script>
        var app = angular.module('MyApp', []);
        app.controller('MyCntl', function ($scope) {
            $scope.guys = ['jeff', 'joe', 'bob'];
        })
    </script>
</head>
<body ng-app="MyApp">
    <ul ng-controller="MyCntl">
        <li ng-repeat="guy in guys">{{ guy }}</li>
    </ul>
</body>
```

Angular 是唯一利用普通 Javascript 对象(pojo)为模型层的主要前端框架。这使它非常容易与现有的数据源集成和基本数据共存。

比方说，你做一个 AJAX 调用从一个 API 获取一些数据。在可以绑定数据到文档对象模型时，大多数的框架需要你来包装有 getter 和 setter 模型对象中的数据。 THA getter / setter 的 non-Angular 框架是如何传播数据变化事件的方法。

Angular 通过使用一种叫做脏检查的方法来解决这个问题，其中随着时间的推移，对数据的快照进行比较，看看是否有什么有改变。然而以这种方式，过程中当然也有一些缺点（例如范围，应用，

数据绑定限制等）。

3. 依赖注入模型：

```
var app = angular.module('MyApp', []);

app.factory('changeUrlTo', function ($location) {
    return function (destinationUrl) {
        $location.path(destinationUrl);
    };
});

app.controller('MyCntl', function (changeUrlTo) {
    changeUrlTo('/another-page');
});
```

有一些人喜爱、依赖它而有一些人讨厌它。如果你打算用 Angular 工作，你需要坚定地使用依赖注入。我个人喜欢它，因为它具有强大的单元测试功能，并促进了代码模块化。

单元测试前端代码通常是困难的，因为有很多互相依赖的模块。Angular 的依赖注入可以让你模拟出许多依赖关系相互隔离的单个组件。

AngularJS 的功能

功能 1：双向数据绑定

把你的模型想象为你应用程序的唯一来源。你的模型是在你的应用程序中读取或更新任何内容的地方。

数据绑定可能是 AngularJS 最酷，最实用的功能。这将节省你编写样板代码的大量时间。一个典型的 Web 应用程序可能包含的代码库的高达 80%，专门用于遍历，操作和监听文档对象模型。数据绑定使这个代码消失了，这样你就可以专注于应用程序的开发。

假设你的模型是你的应用程序的单一可信来源。使用你的模型，你可以去读或更新任何在您的应用程序。数据绑定指令提供 projectionof 模型的应用程序视图。这个视图是无缝的，没有任何的影响。

传统上，当模型改变时，开发人员负责手动操纵文本对象模型元素和属性，以反映这些变化。这是一条双行道。在一个方向上，模型变化驱动文本对象模型元素的变化。另外，文本对象模型元素变化需要在模型中进行更改。这是进一步复杂化的用户交互，因为开发人员负责解释的交互，把它们融合成一个模型，并更新视图。随着应用程序的规模和复杂性的体现，这是一个非常死板和繁琐的过程，这就很难控制。

必须有一个更好的方法来解决这些问题。AngularJS 的双向数据绑定处理文本对象模型和模型之间的同步，反之亦然。

下面是一个简单的例子，它演示了如何绑定的输入值到<h1>元素。

```
<!doctype html>
<html ng-app>
    <head>
        <script src="http://code.angularjs.org/angular - 1.0.0rc10.min.js">
        </script>
    </head>
    <body>
        <div>
            <label>Name:</label>
            <input type="text" ng - model="yourName" placeholder="Enter a name
    here">
                <hr>
                <h1>Hello， {{yourName}}!</h1>
        </div>
    </body>
</html>
```

这是非常简单的设置，但是具有奇效。

功能 2:模板

最重要的是认识到，任何时候都可以用 AngularJS 操作模板字符串。这是所有浏览器共同的的文本对象模型。

在 AngularJS，模板只是常规的 HTML。 HTML 词汇扩展，包含在模型上应该如何投射到视图的说明中。

HTML 模板是由浏览器解析成文本对象模型。文本对象模型成为 AngularJS 编译器的输入。AngularJS 遍历文本对象模型模板呈现指令，这被称为指令。总的来说，指令负责设置应用程序视图的数据绑定。

要认识到，在任何时候 AngularJS 操作模板字符串都是很重要的。输入到 AngularJS 的是浏览器的文本对象模型，而不是一个 HTML 字符串。数据绑定是文本对象模型的转换，而不是字符串连接或 innerHTML 的变化。使用文本对象模型作为输入，而不是字符串，是 AngularJS 与其兄弟框架最大的分化。使用文本对象模型的是什么允许扩展指令词汇和建立自己的指令，甚至是抽象的它们变成可重复使用组件！

这种方法最大的优点之一是，它会创建一个紧密的设计人员和开发人员之间的工作流程。设计师通常会把 HTML 作为他们的标记，然后开发人员接力并通过绑定和少许的努力即可实现功能。

这里是一个例子，我使用 ng-repeat 指令数组遍历图像和填充。它本质上是一个 img 模板。

```
function AlbumCtrl($scope) {
    scope.images = [
        {"thumbnail":"img/image_01.png",  "description":"Image 01 description"},
    {"thumbnail":"img/image_02.png",  "description":"Image 02 description"},
    {"thumbnail":"img/image_03.png",  "description":"Image 03 description"},
    {"thumbnail":"img/image_04.png",  "description":"Image 04 description"},
    {"thumbnail":"img/image_05.png",  "description":"Image 05 description"}
    ];
}
<div ng - controller="AlbumCtrl">
    <ul>
        <li ng - repeat="image in images">
            <img ng - src="{{image.thumbnail}}" alt="{{image.description}}">
        </li>
    </ul>
</div>
```

值得一提的是，这从一个侧面说明， AngularJS 不会强迫你学习一种新的语法或从你的应用程序中提取你的模板。

功能 3： MVC

AngularJS 包含原始 MVC 软件设计模式背后的基本原理以及如何构建客户端 web 应用程序。

MVC，或模型-视图-控制器模式，意味着很多事情对不同的人有很大的区别。AngularJS 没有实现传统意义上的 MVC，而是更接近 MVVM(Model-View-View-Model)。

模型

模型是应用程序中的数据。模型只是普通的 JavaScript 对象。它不需要继承框架类，可以通过包装代理对象，或使用特殊的 getter / setter 方法来访问它。事实上，我们正在处理的 JavaScript 有一个非常好的特性，它减少了应用程序的样板。

视图模型

视图模型是一个对象，提供具体的数据和方法保持特定的视图。视图模型是 AngularJS 应用程序

内的范围对象。范围只是一个很小的 API，旨在检测和广播改变其状态的简单的 JavaScript 对象。

控制器

控制器负责设置初始状态和扩大 scope 的范围与方法来控制行为。值得注意的是，控制器不存储状态，不与远程服务交互。

视图

AngularJS 后存在的观点是 HTML 包括 HTML 解析、编译和绑定，包括呈现标记。

这个部门架构应用程序创建一个坚实的基础。scope 的范围有一个参考数据，控制器定义行为，和视图处理布局和交互转移相应控制器响应。

功能 4:依赖注入

AngularJS 有一个内置的依赖注入子系统，帮助开发人员通过使应用程序更容易开发，理解和测试。

依赖注入(DI)允许你直接获取依赖关系，而不必自己去寻找它。把它看作你只需要说“嘿 X，我需要你”，依赖注入就会把 X 创建和提供给你。

访问核心 AngularJS 服务， 作为一个参数添加服务是很简单的；AngularJS 将检测你需要的服务，并为你提供一个实例。

```
function EditCtrl($scope， $location， $routeParams) {
    // Something clever here...
}
```

你还可以定义自己的定制服务，来完善注入的功能。

```
angular.
    module('MyServiceModule'， []).factory('notify'， ['$window'， function (win) {
        return function (msg) {
            win.alert(msg);
        };
    }]);
function myController(scope， notifyService) {
    scope.callNotify = function (msg) {
        notifyService(msg);
    };
}
myController.$inject = ['$scope'， 'notify'];
```

功能 5:指令

AngularJS 的指令是我个人最喜欢的特性。你是否希望你的浏览器会有新的功能？好了，现在可以！这是 AngularJS 中我最喜欢的一个部分。它也可能是 AngularJS 最具挑战性的方面。

指令可用于创建自定义的 HTML 标记，作为新的自定义小部件。他们还可以采用有趣的方式，用于装饰元素的行为和操作 DOM 属性。

这里是一个简单的例子，用来监听事件的指令和更新其 scope 的范围。

```
myModule.directive('myComponent', function(mySharedService) {
    return {
        restrict: 'E',
        controller: function($scope, $attrs, mySharedService) {
            $scope.$on('handleBroadcast', function() {
                $scope.message = 'Directive: ' + mySharedService.message;
            });
        },
        replace: true,
        template: '<input>'
    };
});
```

然后，你可以使用这个自定义指令。

```
<my-component ng-model="message"></my-component>
```

创建你的应用程序组成的离散组件，使它非常容易添加、更新或删除功能。

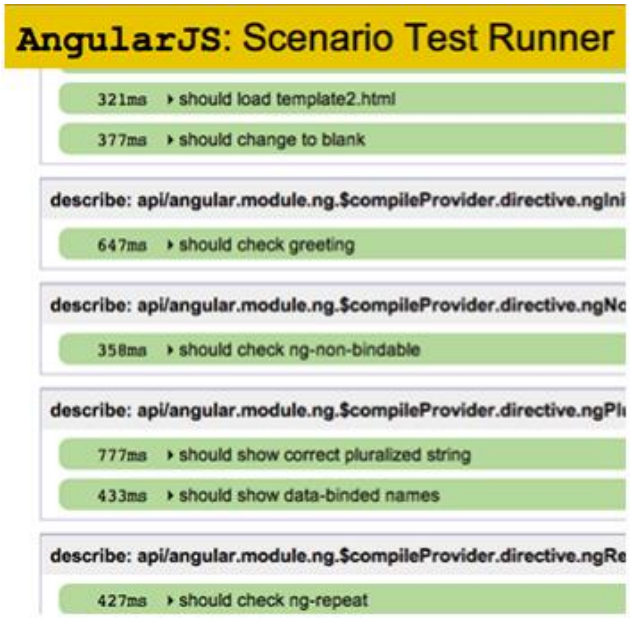
附加功能:测试

AngularJS 团队很强大，因为任何 JavaScript 编写的代码都需要有强大的测试集。他们设计了 AngularJS 的记忆性和可测试性，使得测试 AngularJS 应用程序尽可能容易。那么我们没有理由不这样去做 Angular 的测试。

对于 JavaScript 动态和解释，而不是编译，对于开发者而言，能够坚持编写测试是非常重要的。

AngularJS 完全从头开始编写测试。它甚至还有一个端到端的单元测试运行器的设置。如果你想看到的这个 action，去看看 angular-seed 项目 https://github.com/angular/angular-seed。

一旦你有了种子工程，可以对它运行测试。这里的输出如下:



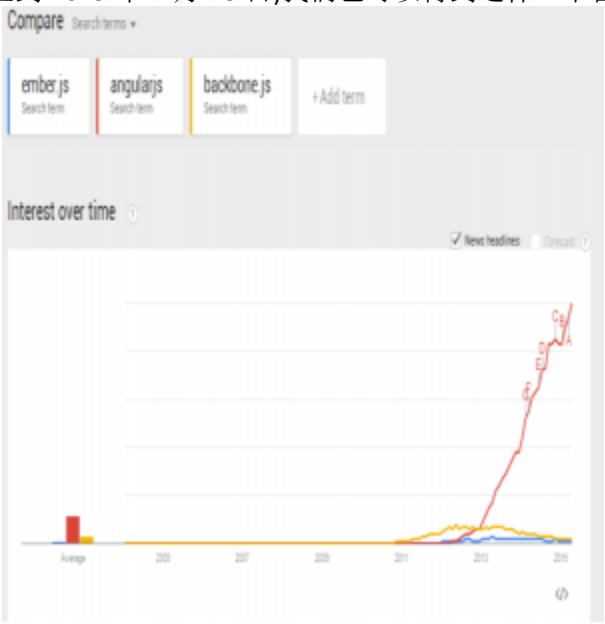
API 文档的端到端的测试，说明某一部分框架的工作方式是一件困难的工作。一段时间后，我发现自己可以直接测试，看它的运行过程，然后阅读剩下的文档，最后完成自己的项目。

社区比较：

社区是选择一个框架时最重要的考虑因素之一。一个大型社区意味着更多的问题得到解答，更多的第三方模块，更多的 YouTube 教程等等。我建立了一个截至 2014 年 8 月 16 日的数据表。作为 github 上关注第六多的项目，在很多方面对比 StackOverflow 、Ember 、Backbone ，Angular 绝对是赢家，如下图所示:

Metric	AngularJS	Backbone.js	Ember.js
Stars on Github	27.2k	18.8k	11k
Third-Party Modules	800 ngmodules	236 backplugs	21 emberaddons
StackOverflow Questions	49.5k	15.9k	11.2k
YouTube Results	~75k	~16k	~6k
GitHub Contributors	928	230	393
Chrome Extension Users	150k	7k	38.3k

然而，所有这些指标只是显示每个框架的当前状态。看哪个框架增长较快是很有趣的事情。幸运的是，使用谷歌趋势(直到 2015 年 4 月 18 日)我们也可以得到这样一个答案：



结论：

angular 用于扩展 HTML 的创新方法可以让 Web 开发人员的灵感源源不断，这具有很大的意义。在大型社区和谷歌的背后，angular 在这里停留和成长，它无论在快速原型项目还是规模化生产应用中都非常好用。

参考文献：

[1] AngularJS vs.Backbone.js vs. Ember.js (https://www.airpair.com/js/javascript - frameworkcomparison)

[2]https://www.airpair.com/angularjs/posts/jqueryangularjs - comparison - migration - walkthrough

[3] http://code.tutsplus.com/tutorials/3 - reasons - tochoose - angularjs - for - your - next - project - - net - 28457