基于思维导图的团队协作系统①

罗俊健, 陈永钊, 张英港, 赖源劲

(华南师范大学 计算机学院, 广州 510631)

摘 要: 思维导图作为一种高效率的可视化工具, 在线下团队协作中的运用也越来越多, 而于在线协作工具的运用并不多. 该文针对思维导图在团队协作工具的应用来进行阐述, 描述了一个采用 B/S 架构, 以思维导图为重要功能模块的团队协作系统的实现. 阐述了其主要功能、系统结构、关键实现技术以及开发过程中相关的工程方法. 基于测试驱动的开发形式和 IOC 机制的运用, 使其具备较好的扩展、维护和二次开发的能力. 而基于思维导图来展现团队协作中各类任务的相互关系, 以图像化交互的形式来管理项目, 与传统的协作系统的线性任务管理相比, 具备更好的交互和较低的使用门槛.

关键词: 思维导图; 团队协作; 单页 Web 应用; SVG; Restfull

Teamwork System Based on Mind Map

LUO Jun-Jian, CHEN Yong-Zhao, ZHANG Ying-Gang, LAI Yuan-Jin

(School of Computer, South China Normal University, Guangzhou 510631, China)

Abstract: Mind map as an efficient visualization tools, its use of online and offline teamwork is also more and more, but the use of online collaboration tools is not much.The paper describes an implementation of a teamwork tool adopting B/S structure, whose significant function moduleis Mind Map. In this paper, the main functions, system structure, key technology and the related engineering method in the development process would be stated. It also talks about how to use TDD-Test-Driven Development and the IOC Vessel to optimize the favorable code organization structure and the ability of extension, maintenance and secondary development would be provided. Contrasting with the linear task management in the traditional collaboration system, the mind map is the visual and nonlinear tool aiming to stimulate thinking as well as brain bloom and flow and would have better interaction and lower threshold if we use the mind map to reveal the mutual relation of the teamwork tasks, and manage the project in the form of graphical interaction.

Key words: mind map; teamwork; single page web applications; SVG; Restfull

1 前言

团队协作工具对于团队运作和管理的重要性已毋庸置疑, 而当中所涉及到的信息传递、任务划分、人员指派、进度把控等方面都需要一个规划化的流程机制来进行管理, 否则工作就容易嵌入到低效, 闭塞, 混乱的困境. 在目前的协作工具中, 文档协作软件如有道云协作、Google Docs、石墨等, 虽然有实时编辑内容与方便的资料管理功能, 但欠缺权限管理, 难度跟踪任务进度, 以文档为核心难以胜任复杂项目; 但像 Teambition、Worktile、今目标等团队协同软件, 尽管它们提供了丰富的功能和权限管理, 但操作复杂, 难以上手; 至于 QQ、微信、skype 等即时通信软件, 虽然具有部分协作功能, 但容易受到干扰, 成果不易保存[1]. 而思维导图的表现形式更符合人脑的思维方式, 用思维导图可以更清晰, 简洁地显示项目结构和任务进度, 对于任务而言起一定的梳理作用. 思维导图最初是 20 世纪 60 年代英国心理学家东尼·博赞(Tony Buzan)发明的一种笔记方法[2]. 东尼·博赞认为思维导图是对发散性思维的表达, 因此也是人类思维的自然功能[3]. 思维导图可以应用于生活的各个方面, 提高学

① 基金项目:大学生创新创业训练计划项目(2014115);国家自然科学基金(61272066)
收稿时间:2016-01-04;收到修改稿时间:2016-02-29 [doi:10.15888/j.cnki.csa.005349]

习能力和思维的梳理能力, 从而提高人们的效率. 因而, 基于思维导图构建一款团队协作工具, 有利于降低工具的使用门槛, 同时也能通过工具的形式为团队协作的运作到带来规范化的、更为高效的工作流, 消除沟通的阻碍, 获得清晰的任务分派管理和简洁的资源管理机制, 使团队协作简单化, 高效化, 条理化. 本系统采用 B/S 架构, 基于浏览器对 SVG 的支持实现在线绘图, 降低部署成本, 而整合版本控制系统的代码依赖管理形式也有利于降低维护和升级的成本.

2　系统组成

2.1　总体架构

本系统采用了 B/S 架构, 单页 Web 应用的实现方式. 单页 Web 应用的核心思路是浏览器用户加载一个单独的 HTML 页面之后, 剩余的使用操作均无需离开此页面. 因而用户不需要频繁地刷新和切换页面, 页面实际上是按需部分更新的, 得益于现代浏览器技术的发展, 利用标准的历史管理 API 来实现功能页的前进后退, 也能利用 GUP 等硬件来提升页面的渲染性能, 使得单页 Web 应用也能拥有比拟本地原生应用的流畅体验. 单页 Web 应用比传统的 Web 实现具备更多优势, 使得应用的状态可以完全掌握在客户端层面, 复杂的逻辑可以在浏览器端的界面层实现而不需要涉及到后端的路由规则[4]. 前端和后端双方主要基于约定的数据协议进行纯数据交互, 能够与面向服务的架构更好地结合.

与传统的 C/S 架构相比, 用户无需安装特定平台的应用程序, 开发者也无需针对各个平台进行单独的开发, 在开发、部署和维护成本上具备一定的优势. 而单页 Web 应用按需获取的特点, 也能减轻后端服务器的压力, 并且使得后端开发的重点转移到功能层的开发上, 表示层的重心则转移到前端开发, 令二者耦合分离.

2.2　功能模块划分

图 1　功能模块

3　主要功能

3.1 可视化的任务管理

3.1.1 使用情景示例

① 用户登陆后, 在项目列表中选择需要进行操作的项目, 进入后在“工作台”页面选择新建任务, 按提示填写相关的任务信息, 并指派相关的任务人员, 如图 2.

图 2　新建任务界面

② 当对某一项目中的任务完成指派后, 可以在“工作台”看到总体的项目任务概况, 如图 3.

图 3　总体的项目任务概况

③ 点击任务概况的思维导图, 还可以切换显示的角度, 如具体显示某一个任务的情况, 此时可以通过项目名右侧的面包屑导航来返回上一级任务, 如图 4.

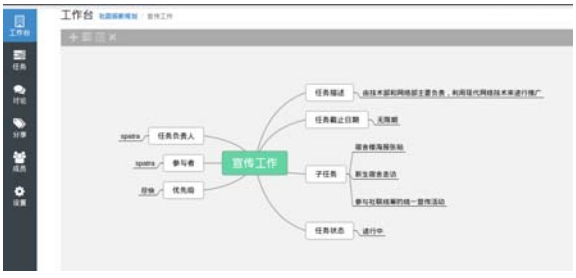

图 4　切换显示角度

④ 当某些任务之间的关系需要调整的时候，可以直接在图上进行拖放操作，例如需要将”拉赞助商“这一任务的重要性提高，进行详细的规划. 则可以直接将”商家赞助“这一任务拖动到项目名“社团招新规划”的上面，从而让其独立，并且为其创建新的具体任务，相应的父子级任务关系会自动实现调整，如下图 5 和图 6.

图 5 直接拖动，创建新的具体任务

图 6 父子级任务关系自动调整

3.1.2 与一般的任务管理工具比较

一般化的任务管理工具，则仅具备如下的任务列表视图(如图 7). 这样的任务视图虽然能够看出具体的任务指派，但只是一个线性的描述，而忽视了任务与任务之间的关联性，某一任务方向可能随着项目的进展而出现拆分的需求则得不到满足. 从数据可视化的角度而言也难以让用户一观全貌.

图 7 任务列表视图

3.1.3 思维导图对大脑记忆能力和学习能力的提高

思维导图呈现的是非线性、连接性、平面性、块状性的视觉图形. 基于大脑对色彩差别的区分能力，思维导图中的节点以颜色进行区分，方便大脑对不同模块进行区分和管理. 另外一点，思维导图的结构和神经网络模型相似. 神经网络和思维导图的基本构成单位称为结点或单元，并且整体结构都具有连接性. 联结主义心理模型以神经科学为基础对人的神经系统进行模拟，比符号主义模型更具 有神经学意义上的合理性. 简单单元联接在一起就具有了复杂的行为和能力，在一定程度 上可以像人脑一样地工作. 简单化的单元组织在一起产生了许多有趣的特性，如表现出“内容地址”式记忆、模糊或部分的刺激可以提取整个网络中的记忆痕迹等. 如果网络的一部分受损伤，也会如人脑那样衰退，记忆成绩随着损伤的程度而逐渐下降，而不是一次性的崩溃[5]. 联结主义认为，学习就是 不断优化脑中的知识网络[6]. 因此，在思维导图绘制的过程就是对大脑知识和想法进行 “碎片整理” 并“不断优化”的过程. 可以认为，思维导图利于促进知识体系的完善，因此也利于促进协作管理的优化和完善.

3.2 自动化的消息通知

在团队协作中，消息共享与互通的重要性是毋庸置疑的. 项目的管理者和执行人都需要获取当前项目中各类活动的最新动态，才能根据当前的进度来安排或调整当前的工作. 在传统的团队协作中，成员可能会通过电话或 QQ 等即时聊天工具来交换信息，又或者需要采用群发邮件的方式来进行相互沟通. 此类方式从单纯的沟通而言是足够的，但缺乏相应的智能和自动化，成员需要自行决定和管理那些消息需要通告，并为此执行具体的操作.

在本协作系统中，项目中各个任务的改动、成员的分享动态、议题的讨论进度等各类变动，当前端与后端进行数据同步的时候，后端系统会监控数据改动，并为此自动生成相应规格的通知，从后端推送给相关联的成员. 如果某一成员在使用系统的时候接收到活动推送，系统则会向当前用户发出提示，该提示包含了声音提示和页面通知，如下图 8，点击后即可进行阅读. 如果用户当前没有使用系统，则在下次登陆时会有相应的未读信息提示，并可进入收件箱页面进行阅读，如下图 9. 这些通知中附带了的相关资源链接，点击后还可以跳转到具体的栏目.

图 8 通知提示

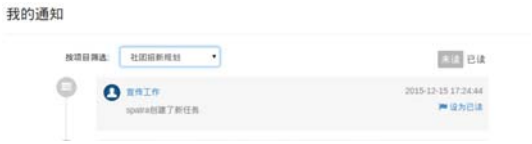

图 9 收件箱页面

4 关键技术

4.1 基于 SVG 实现的前端绘图

SVG(Scale Vector Graphic), 是基于可扩展标记语言(标准通用标记语言的子集), 用于描述二维矢量图形的一种图形格式. 此格式的标准是一种开放标准, 并由万维网联盟所制定, 因此 SVG 在 Web 开发中广泛应用的 HTML/CSS 有着良好的集成. 与另一个在 Web 前端开发中所广泛应用的 Canvas 方案相比, SVG 具有基于形状而非像素、能通过 CSS 进行修饰、能与 DOM 事件系统良好结合, 无损缩放等优点, 而与 Flash 和 Silverlight 等需要浏览器扩展支持的技术相比, SVG 则具有浏览器原生支持、更为稳定和高效的特性. 基于 SVG 实现的绘图技术, 具有对象化和易于交互的特点, 适合用于思维导图在线绘制的实现.

在本系统的思维导图中, 实现的主要功能点有: 添加节点, 删除节点和拖动节点. 这些功能点需要用到以下技术点:

① 绘制图形. SVG 有一些预定义的形状元素, 如矩形、线、路径等形状, 使用这些形状元素即可绘制思维导图的图形: 节点和连线. 一个节点可由一个 SVG 文本元素以及一个包裹着文本的矩形组成, 要保证文本在矩形的中央位置. 为了美观, 根节点与第一层节点的连线使用贝塞尔曲线, 其他连线使用直线, 使用 svg 的路径元素即可绘制贝塞尔曲线和直线.

② 添加和删除节点时节点位置的渲染. 当添加一个节点时, 要保证过父节点中心点的水平直线垂直平分当前节点与同级节点的高度所形成的线段, 这就需要移动当前节点与同级节点的垂直位置. 父节点的同级节点的垂直位置也要改变, 比父节点高的节点向上移动, 比父节点低的节点向下移动. 删除节点的节点位置改变操作与添加节点相似.

③ 拖动节点: 在此绘图模块中, 节点的拖动分为根节点拖动和非根节点拖动. 根节点的拖动会导致关联子节点位置的变动, 在 Canvas 等非矢量点阵绘图体统中, 实现根节点的拖动需要对涉及到的所有的图形坐标进行重新计算, 从而带来相当的运算量. 在基于 SVG 实现的本模块中, 仅需要对 SVG 的视野位置进行修改, 即可实现变动. 而拖动非根节点用于改变节点的父子关系, 节点拖动结束后, 如果该节点与另一个节点重叠, 则使重叠节点变为该节点的父节点, 否则, 节点回到原来的位置.

4.2 符合 RESTFull 架构的前后端交互协议

RESTFull 架构是符合 REST 原则(Representational State Transfer, 表现层状态转化)原则的架构. REST 提供了一组架构约束, 当作为一个整体来应用时, 强调组件交互的可伸缩性、接口的通用性、组件的独立部署、以及用来减少交互延迟、增强安全性、封装遗留系统的中间组件[7],与只是简单使用 GET/POST 和随意定义 URI 的应用相比, RESTFull 使得前后端交互变得高性能和规范化.

在本系统中, 我们基于 RESTFull 架构的里面, 结合实际情况实现了一套前后端交互的 URI 接口, 其基本定义如下.

资源	GET	PUT	POST	DELETE
组资源的 URI, 比如 http://mindmap.com/resources/	列出 URI, 以及该资源组中每个资源的详细信息(后者可选).	使用给定的一组资源替换当前整组资源.	在本组资源中创建/追加一个新的资源. 该操作往往返回新资源的 URL.	删除整组资源.
单个资源的 URI, 比如 http://mindmap.com/resources/142	获取指定的资源的详细信息, 信息的格式可由自由设定.	替换/创建指定的资源. 并将其追加到相应的资源组中.	把指定的资源当做一个资源组, 并在其下创建/追加一个新的元素.	删除指定的元素.

所谓的资源, 具体到本系统, 则是项目、任务、项目成员等数据集合的抽象体. 前端的 Web 应用使用合适的 HTTP 动词来对资源进行操作, 从而获取或修改数据. 以请求项目 1 的数据为例, 对资源的获取过程进行描述:

① 前端 Web 应用基于 HTTP, 通过 GET 方法访问 URI: http://tmindmap.net/project/1; ② 后端服务器根

据请求和预设的路由规则, 调用控制器类对象 ProjectController 的 show 方法来处理该请求; ③ 在 ProjectController::show 方法中, 根据请求 id(本例中是 1)来执行数据库查询, 如有匹配则将结果数据用 JSON 序列化后存到 HTTP 响应体中, HTTP 响应码设置为 200; 如无匹配则 HTTP 响应体为空, 且状态码为 404; ④ 前端 Web 应用得到 HTTP 响应后, 对响应体中的 JSON 形式的数据进行反序列化, 从而得到最终的数据.

如果是其他的操作, 例如删除或修改资源, 则请求 URI 不变, 但 HTTP 方法分别采用 DELETE 和 PUT. 而且资源之间还可以嵌套使用, 如 http://tmindmap.net/project/1/members 则可以获取某个项目下的所有成员的信息. 采用 JSON 作为数据序列化方案, 并基于 JSON Schema(基于 JSON 来描述数据体的结构的一套开源协议)来描述某一接口下数据交换的具体约定的优点在于 Web 开发领域中 JavaScript 的广泛使用而具体良好的兼容性, 并且序列化结果比 XML 或二进制格式具备更好的可读性, 方便调试开发.

4.3 基于事件驱动的消息生成

在本系统中, 对数据所进行的增、删、查、改等操作, 是基于相应的 ORM 模块来实现的, 对存储在数据库中的各项数据操作会反映在相应对象的 create、delete、query 等方法的调用上. 而 ORM 模块中实现了一套事件机制, 对于业务开发而言, 侧重点在于如何处理发生的事件而不需要知道事件是如何产生的, 事件处理模块和业务逻辑模块保持良好的隔离性. 实现关键点描述如下: ① 统在初始化的时候, 会创建并维持一个 Event 类对象(单例), 该类实现了事件的发送和监听的具体方法; ② 负责业务逻辑的各类对象均可以向全局的 Event 类对象的 listen 方法传入事件名和实现了 EventCallback 接口的操作对象来实现对某一事件的监听; ③ 当用户通过 ORM 模块进行数据操作时, 各类操作均生成相应的描述信息, 并通过全局的 Event 类对象的 fire 方法来创建并传递具体的事件对象; ④在消息生成模块的逻辑操作对象中对任务更改、成员加入等操作进行监听, 并在操作对象中将相关数据写入用户的待通知表中; ⑤当用户登录后, 系统查询待通知表中是否有新的记录, 如有则生成新的消息并下发给用户.

在本系统中, 通过编写相关的事件处理程序代码, 来得知用户的各项操作变动, 并通过一定的自动化筛选策略和消息模板来生成最终的具体通知消息. 从而实现了同一项目或任务中某一成员的操作或进度更改, 会自动地告知相应的干系人. 这一步骤是无需用户干涉的, 并且对应的逻辑改动也不会影响到用户的操作本身.

4.4 采用 MVVM 架构的前端代码组织形式

对于单页 Web 应用的实现方式下的应用开发, 由于将大量的用户交互逻辑从后端服务处理转移到前端浏览器控制, 使得前端代码不再是可有可无的动态效果实现, 而是整个应用正确运行的关键. 因此前端代码在此开发模式下, 也变得越来越复杂, 像 MVC 等传统的应用开发模式也被引入到前端开发中, 用于组织越来越复杂的前端代码.

在传统的 MVC 架构中, 应用的组织形式如图 10 . 具体到 Web 前端开发, 则是通过 Ajax 等方式从后端服务器获取到数据之后, 生成或封装对应的 Model, 然后调用对应的模板函数或类方法将其渲染到 HTML 页面上. 页面的 URI hashCode 的变动或 DOM 事件则是 controller, 用于触发相应的业务逻辑, 或变更数据, 或重新获取数据.

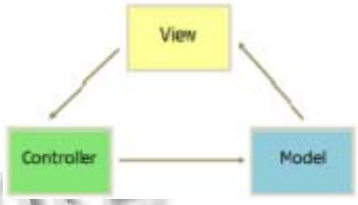

图 10 传统 MVC 架构的应用组织形式

而在 MVVM 架构中(如图 11), 业务逻辑不用再关注数据如何呈现到页面, 而由框架更新 Model 和 View, 从而得到了所谓的双向数据绑定. 将数据交给 VM 之后, VM 自行更新视图, 而由 DOM 事件触发的视图改动若涉及到数据, 则所关联的 Model 也会被更新. 在本系统中, 我们选择了 Google 主导和维护的 Web 前端开发框架 AngularJS 是负责这部分的框架逻辑. 基于此架构实现的单页 Web 应用, 对于日趋复杂的用户交互逻辑, 通过双向数据绑定的抽象特性, 可以复用大部分的细节处理代码. 因此, 使用 MVVM 模式可以将视图逻辑和业务逻辑分开, 在保持业务逻辑不变的情况下, 只要针对视图逻辑进行不同的设计即可[8].

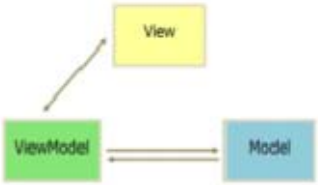

图 11 MVVM 架构的应用组织形式

4.5 对象管理与测试驱动开发

在采用面向对象设计实现的系统中, 随着开发的进行, 类与类之间的关系日益复杂, 对象之间如何处理依赖关系成为必须解决的问题. 而 IoC 全称 Inversion of Control , 即控制反转, 是 Apache Avalon 项目创始人之一 Stefano Mazzocchi 提出的, 用来解决组件之间依赖关系 、配置及生命周期的设计模式 , 其中对程序中的组件依赖关系的处理是 IoC 的核心 [9]. 在本系统中, 后端开发和前端开发中均具备了一套 IoC 容器机制, 使用者通过定义工厂方式的形式来实现相关对象的构造方式, 并通过统一的工具类方法来获取具体对象的实例. 从而使得本系统的各部分模块能以松耦合和一致的形式来获取所需要的各类对象, 为测试驱动开发形式下的单元测试的实现打下了基础.

测试驱动开发(Test Driven Development , TDD) 作为 XP 编程思想的一种主要实践 , 可以有效地让程序开发人员开发出更高品质的、经过完整测试的程序 [10]. 区别于传统的软件开发模式, 测试先行将更重视测试在整个软件开发过程中的作用并促进项目的进行. 本系统的研发过程使用测试驱动开发, 即通过编写代码的测试用例, 对其功能的分解、使用过程、接口都进行了设计, 然后再完善功能实现. 而且这种从使用角度对代码的设计通常更符合后期开发的需求. 系统基于 PHPUnit 实现自动化的单元测试, 结合 Laravel 中的相关工具类和 IOC 机制, 实现具体的测试用例, 共编写了 127 个测试, 超 800 个断言.

5 总结

基于 B/S 架构实现的团队协作系统, 除具备一般协作管理系统所具备的任务管理、人员指派、人员管理、项目讨论等功能之外, 还结合了思维导图的特性, 提供了思维导图在线绘制及绘图与任务管理的整合功能. 使得用户能够从视觉上对项目的整体进展和划分有比较清晰的了解, 与人的习惯思维所对应. 与传统的协作系统的线性任务管理相比, 具备更好的交互和较低的使用门槛. 而基于业界已广泛使用和验证的 LAMP 构建, 使得系统具备较好的稳定性和安全性. 系统基于测试驱动的开发形式, 基于 IOC 容器机制整合多个模块, 以及良好的代码组织结构, 使其具备较好的扩展、维护和二次开发的能力.

参考文献

1 刘睿,徐翔.在线协作工具在 MCSCL 中的应用研究.软件导刊,2015(12):92–94.

2 Buzan B, Buzan T. The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential, New York: Dutton, 1994.

3 赵国庆.概念图、思维导图教学应用的若干重要问题的探讨.电化教育研究,2012(5):78–84.

4 Kokkonen J. Single-page Application Frameworks in Enterprise Software Development [MS Thesis], Finland: JAMK University, 2015.

5 王益文,张文新.联结主义神经网络及其在心理学中的应用.心理科学进展,2001,(4):368–375.

6 刘菊,钟绍春.网络时代学习理论的新发展——连接主义.外国教育研究,2011(1):34–38.

7 Fielding RT. Architectural Styles and the Design of Network-based Software Architectures [Ph.D. Thesis]. University of California, Irvine 2000.

8 何鹏举,吴飞.基于 MVC 和 MVVM 的天地图架构模式研究.海峡科学,2015(8):13–15.

9 娄锋,孙涌.轻量级 IoC 容器的研究与设计.计算机技术与发展,2007,17(1):91–93.

10 诸剑俊.浅析测试驱动开发.科技创新与应用,2014,(2):51–52.