图易前端设计

技术架构、模块划分、数据接口、约定

一、目录

* 该设计文档不包括数据托管中心和分析平台

二、背景

1. 运行环境

浏览器：Chrome/Firefox/Safari/IE
操作系统：Windows、Mac、iOS

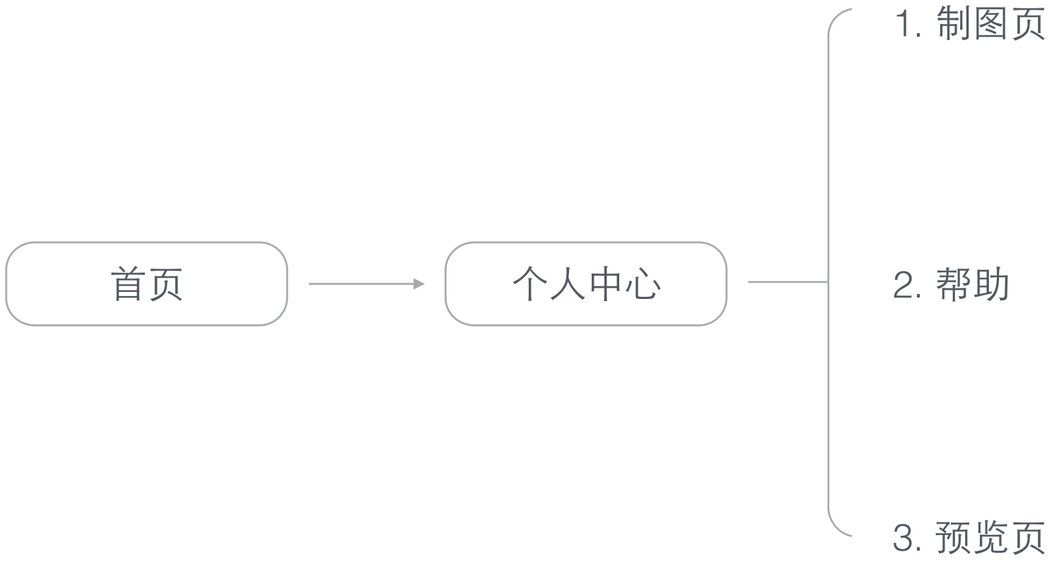
2. 开发环境

浏览器：Chrome/Firefox（Safari/IE在后续适配中会用到）
编辑器：Sublime、Vim
操作系统：Windows、Mac

3. 历史参考

图易4.0、STH、Fishbone、YCharts

三、页面结构



四、功能模块（细分共14个）

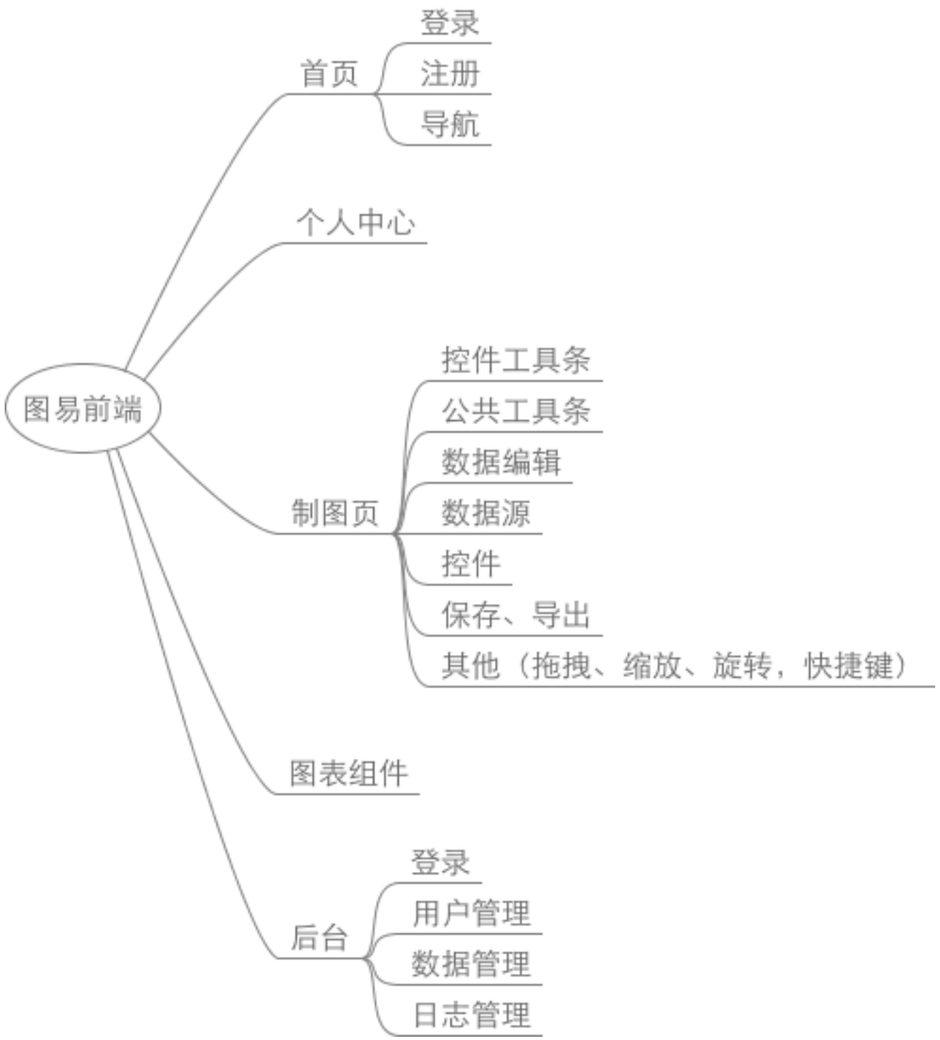
按页面划分

1. 首页（登录、注册、全局导航）
2. 个人中心
3. 制图页

控件工具条、公共工具条、数据编辑、数据源、控件、保存/导出，
其他（拖拽、快捷键、缩放、旋转）

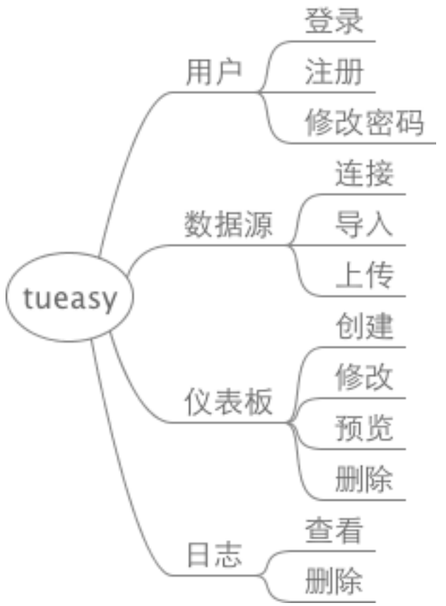
4. 图表组件
5. 后台

登录、数据管理、用户管理，日志管理



围绕数据划分

1. 用户功能
2. 数据源
3. 仪表板
4. 日志



user
id
nickname
email
password
group

以用户划分功能及访问权限

email是唯一字段，用于登录。未登录的用户只能停留在首页

group用于划分用户组，管理员有进入/admin页面的权限

几乎所有数据，都与用户相关

datasource
id
address
name
port
type
description
password
uid
is_import
is_update

用于存储外部数据源，数据源归具体用户对象所有。有明确的外键关系

type用于标识数据源的类型，目前分为MySQL、Oracle和SQL Server

name和description是帮助用户识别的非关键字段

is_import表示数据是否导入，如果导入，需要在数据仓库和数据源表间建立关系

is_update标识数据是否会实时更新，实时数据源会尝试使用socket连接

upload
id
uid
type
file

记录上传的文件，数据表只记录映射路径，上传的内容按照物理文件存储

uid关连user表，上传的内容同样归具体用户所有

type用于标识文件类型，区分图片和数据。但具体的文件解析和存储路径仍要按扩展名区分

file记录文件的物理路径

dashboard
id
type
uid
title
description
createTime
lastModify
content
templateId

记录用户制作的仪表板

uid关连user表，内容归具体用户所有

type用于标识仪表板的类型，是否是模板

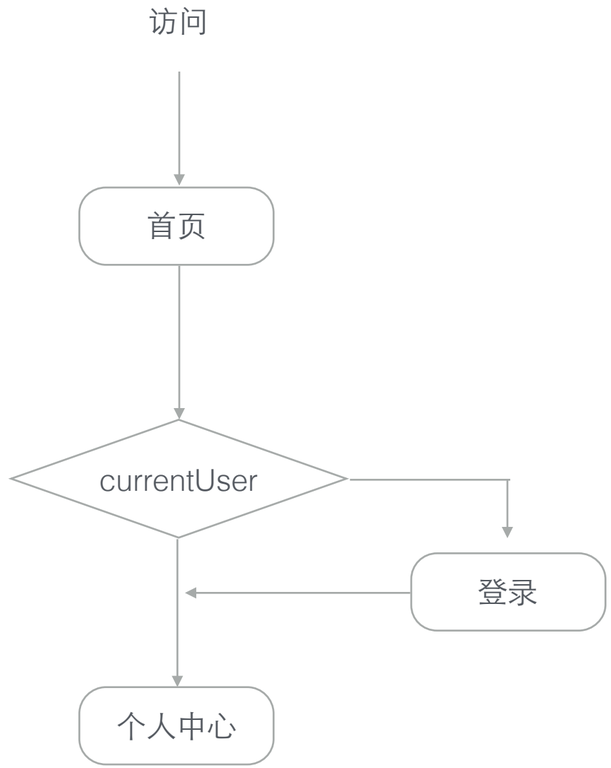
templateId指向类型为模板的仪表板

content内记录了仪表板的全部内容，该字段保存一个JSON字符串。由前端生成、解析、使用，只需原样记录

1. 首页（登录、注册、全局导航）

- 作为整个网站的骨架
- 当页功能包括登录、注册、为其他页面提供持久航、通知、加载进度条等功能
- 登录、注册、获取当前用户与后端有数据交换，通知可能与后端有双向数据交换
- 进入网站必经的页面，所有页面都会用到此模块，所以要注意避免冲突
- 页面代码结构尽量扁平，便于其他模块扩展

- 涉及到的数据接口：user的get、post
  currentUser的get、delete



user、currentUser

键	类型	描述
id	int	id
nickname	varchar	昵称（自动截取）
email	varchar	邮箱，用于登录，唯一
password	varchar	密码
group	varchar	用户组

```
{
    “id”: “1”,
    “nickname”: “sunken”,
    “email”: “luankefei@hiynn.com”
    “password”: “123456”
    “group”: “admin”
}
```

访问任何页面都会尝试调用currentUser获取当前用户状态

当点击登录时，会调用get user。点击注册时，调用post user

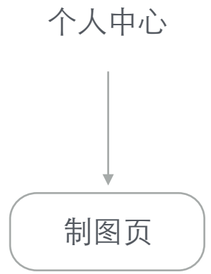
validateMail

键	类型	描述
email	varchar	邮件

```
{
    “email”: “admin@admin.com”,
}
```

注册过程中，会调用validateMail进行邮箱存在验证

2. 个人中心（已完成的仪表板）



- 登录后直接进入的页面，如果是已经登录过的用户，直接越过首页
- 功能包括显示已完成的仪表板，并可进行编辑、预览、重命名和删除操作
- 还可以创建仪表版，套用系统预先存在的模板
- 仪表板分为自己的和系统的（原模板）
- 制图页的唯一入口

- 涉及到的数据接口：dashboard的post、delete、put、get

dashboard

键	类型	描述
id	int	id
type	int	0 - user、1 - autosave、2 - template
uid	int	用户id
title	varchar	标题
description	varchar	描述
createTime	datetime	创建日期
lastModify	datetime	最近修改
content	text	画布的json
tid	int	模板Id

{
“id”: “1”,
“type”: “0”,
“userId”: “0”,
“title”: “只是一个图表”,
“description”: “TODO”,
“createTime”: “2015-01-01 13:12:11”,
“lastModify”: “2015-01-01 13:12:11”,
“content”: “{}”,
“templateId”: 2
}

3. 制图页——数据源

- 数据源的连接，包括MySQL、Oracle和SQL Server
- 从文件导入数据，Excel和CSV
- 数据也可以直接导入到托管中心，再从数据托管中心读取到前端
- 将数据字段展示在左侧列表上

- 涉及到的数据接口：datasource的get
  datatable的get
  upload

datasource

键	类型	描述
id	int	数据源Id
address	varchar	连接地址
name	varchar	数据库名
port	int	端口号
type	int	数据源类型 0 - mysql, 1 - sql server 2 - oracle
description	varchar	数据源描述（用户使用）
password	varchar	连接密码
uid	int	用户id

```
{
    “id”: “1”,
    “userId”: “1”,
    “address”: “102.16.2.5”,
    “port”: “3306”,
    “password”: “123456”,
    “type”: “0”,
    “description”: “测试用的数据源”,
    “name”: “tueasy”
}
```

在制图页面创建数据源连接时，会调用post datasource。

在制图页对仪表板进行二次编辑以及预览页面，可能会调用get datasource

datatable

键	类型	描述
id	int	数据源Id
name	varchar	表名

```
{
    “id”: “1”,
    “name”: “user”
}
```

在制图页面创建数据源连接时，会调用post datatable。

在制图页对仪表板进行二次编辑以及预览页面，可能会调用get datatable

upload

键	类型	描述
uid	int	用户Id
type	int	文件类型，1 - 数据源、0 - 图片
file	text	文件

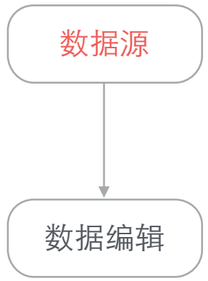
```
{
    “userId”: “1”,
    “type”: “0”,
    “file”: “base64/”
}
```

该接口提供文件传输，返回文件数据。没有再次获取文件的接口

在制图页面使用图片控件时，可能会调用post upload

在制图页面上传本地数据时，可能会调用post upload

4. 制图页——数据编辑

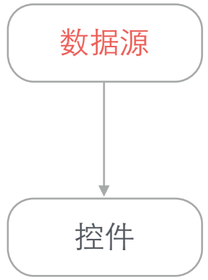


- 类似excel表的编辑
- 类sql的表连接和类excel的公式处理
- 处理过的数据要加入左侧的维度列表

- 涉及到的数据接口：datasource的get
- 依赖两个插件：excel表格、excel公式处理

5. 制图页——控件

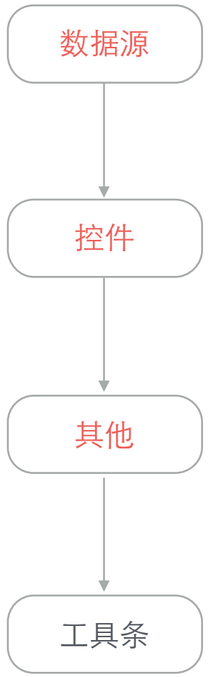
- 类似Visual Studio的控件
- 控件包含文本、图片、线条、复选框、单选框、下拉列表和滑动条
- 控件依赖数据，每种控件有固定的操作绑定逻辑
- 每种控件有独立的工具条



6. 制图页——控件工具条

- 类似iPresst的工具条
- 每种控件有独立的工具条
- 工具条跟随控件移动
- 工具条按照选中控件初始化状态

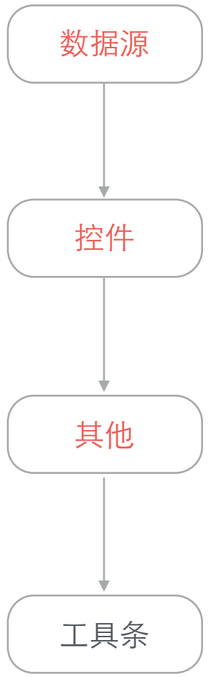
- 必须选中了控件才能工作



7. 制图页——公共工具条

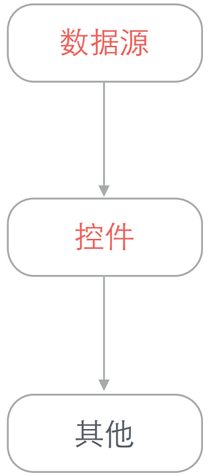
- 类似Piktochart的工具条
- 所有控件公用一个工具条
- 工具条按照选中控件初始化状态

- 部分功能必须选中了控件才能工作



8. 制图页——其他

- 所有控件公用的交互功能，如拖拽、快捷键、缩放、旋转
- 对画布（仪表板）的操作，如变更尺寸



9. 制图页——保存/读取

- 对画布内所有状态进行保存
- 保存分为画布、控件、图表以及控件和图表的相关数据

- 涉及到的数据接口：dashboard的get、put、post
  download

download

键	类型	描述
id	int	模板Id
type	int	0 - html、1 - png、2 - pdf

```
{
    “id”: “1”,
    “type”: “0”,
 }
```

在制图页面点击下载按钮时，可能会调用post download

在预览页面也有可能进行下载，同样调用post download

10. 图表组件

- 整合已有的图表
- 简单、风格一致的接口
- 可扩展，增加、删除图表和组件只修改配置文件
- 可重写，方便二次开发，满足定制化需求
- 可定制，开放图表的全部配置
- 规范代码，加强可读性

- 涉及到的数据接口：图表的数据可能是实时数据源，socket、comet

11. 后台——数据管理

- 对数据源的增删改查，测试数据接口

- 涉及到的数据接口：datasource的get、post、put、delete

12. 后台——用户管理

- 对用户的增删改查，测试数据接口
- 涉及到的数据接口：user的get、post、put、delete

13. 后台——日志管理

- 对日志的增删改查，测试数据接口

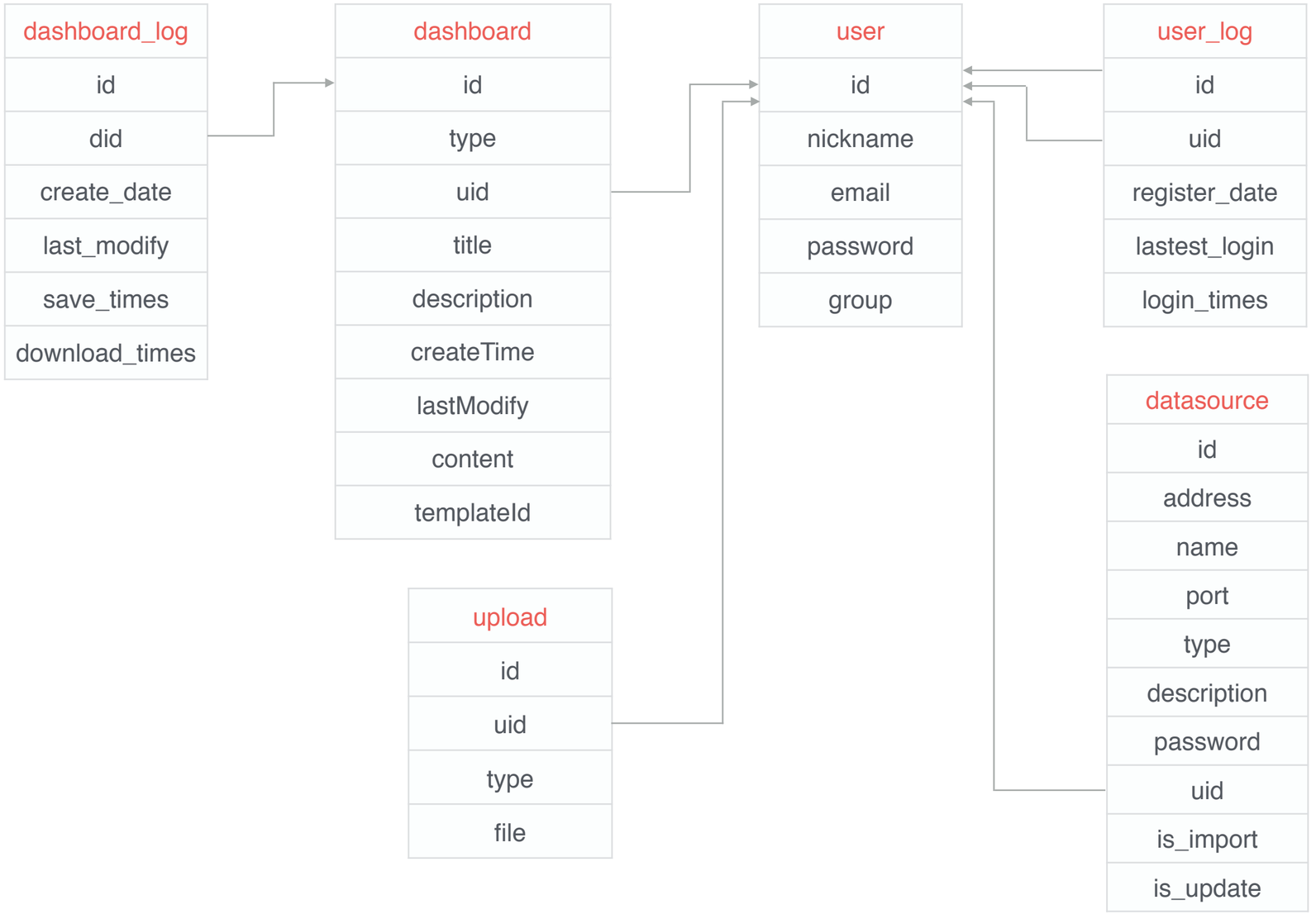
- 涉及到的数据接口：log的get、post、put、delete

14. 后台——模板管理

- 对模板的增删改查，测试数据接口
- 涉及到的数据接口：dashboard的get、post、put、delete

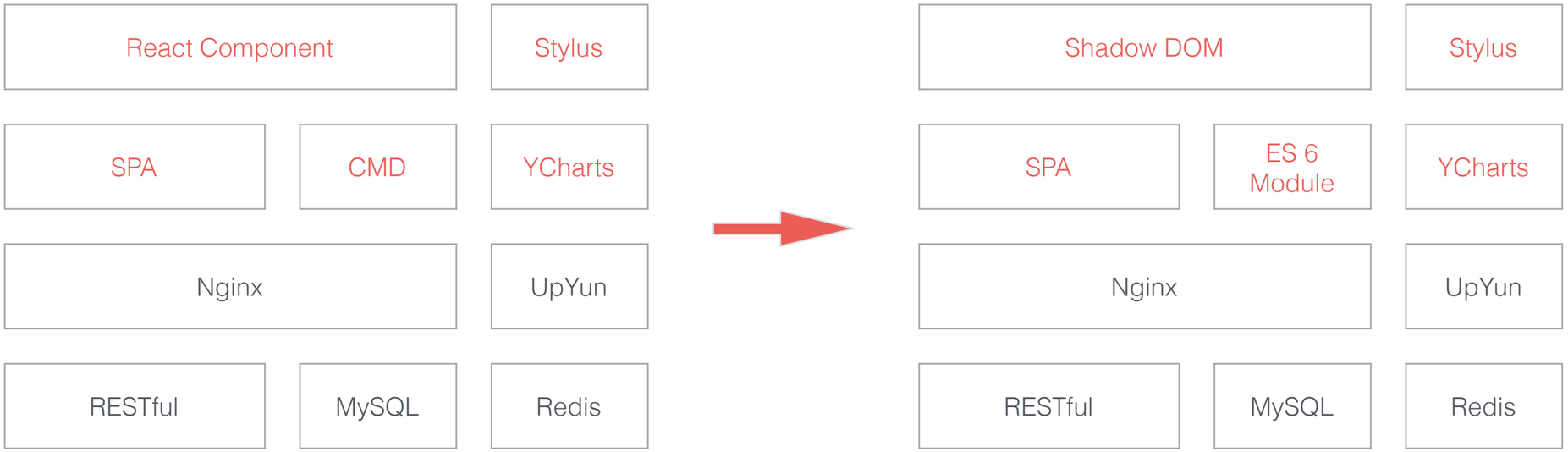
五、数据接口

1. user
2. validateMail
3. dashboard（autosave）
4. datasource
5. datatable
6. upload
7. download
8. log
9. currentUser
10. logout

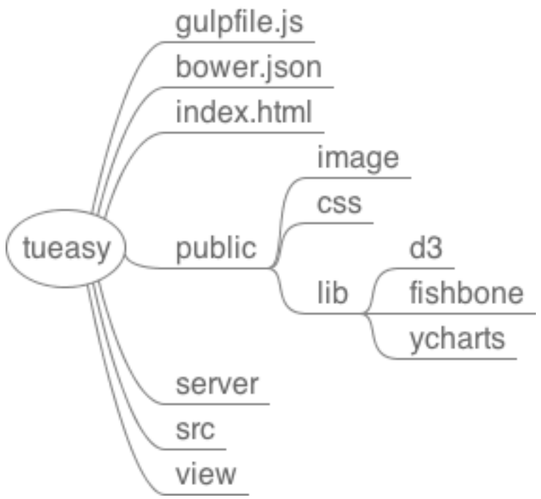

dashboard_log
id
did
create_date
last_modify
save_times
download_times
dashboard
id
type
uid
title
description
createTime
lastModify
content
templateId
user
id
nickname
email
password
group
user_log
id
uid
register_date
lastest_login
login_times
upload
id
uid
type
file
datasource
id
address
name
port
type
description
password
uid
is_import
is_update

六、技术架构

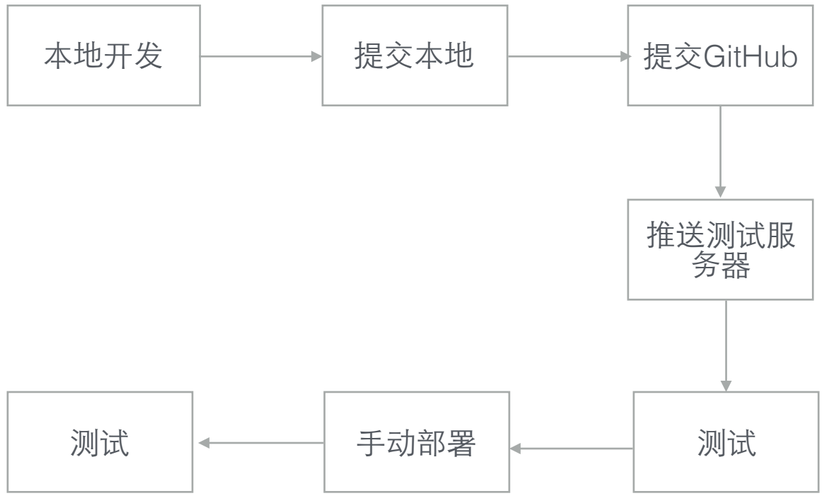
由下至上表示服务器端 -> 客户端的关系



文件结构


tueasy
gulpfile.js
bower.json
index.html
public
image
css
lib
d3
fishbone
ycharts
server
src
view

开发 / 测试


本地开发
提交本地
提交GitHub
推送测试服务器
测试
手动部署
测试

1. RESTful + SPA

优势

- 前后端彻底分离
- 单独测试、单独开发
- 充分利用HTTP，接口优雅
- 体验更佳，可扩展性更强
- 踩着尸体前进，更可靠

不足

- 代码如不规范，容易引发全局污染
- 路由不能兼容低版本浏览器 （IE 8以下）
- 前端工作量增加

2. 模块管理/加载 Seajs

优势

- 大规模使用，相对稳定
- API优雅，减少回调
- 语法更熟悉

Github：https://github.com/seajs/seajs

Seajs是基于CMD规范的前端加载框架，用于解决按需加载和各个JS文件间的依赖

比RequireJS语法更优雅，我们也更熟悉。CommonJS显然不适合在客户端使用

```
// 全局的模块调用API，用于激活入口模块
seajs.use('./index', callback(module) {

    module.init()
})
```

```
// 定义模块
define(function(require, exports, module) {

    var init = function() {

        // doSomething...
    }

    exports.init = init
})
```

```
// 请求模块
define(function(require, exports, module) {

    var a = require('./a')

    a.init()
})
```

3. 开发框架 Fishbone

优势

- 量身定制，完全满足需求
- 类jQuery的语法，学习成本低
- 清楚源码，纠错成本低
- 不兼容IE 6 - 8，更高效
- 接口严格，代码风格统一

不足

- 测试不充分
- 文档不完善
- 部分模块没有开发完成（动画、组件）
- 不能使用jQuery插件

Github：https://github.com/luankefei/fishbone

我们开发的SPA框架，API设计尽量接近jQuery，以降低学习成本

主要用于前端路由和统一过去混乱的代码风格。组件模块规范了通用代码段写法，加强了可复用性

```
// Node模块，非常接近jQuery
$('header').css('height', '100px')
        .css('width', '100px')
        .attr('id', 'header')
        .addClass('header')
```

```
// 事件模块，非常接近jQuery
$(document).on('click', function() {

    console.log('clicked')
})

// 扩展了事件类型，比jQuery更加强大
$('#chart').on('drag', function(e) {

    // dragging
}, function() {

    // drag end
})
```

```
// 请求模块，接近jQuery，支持HttpRequest 2.0，更强大
$.ajax({
    url: '/user/' + id,
    type: 'get',
    before: function() {
        // 请求即将发出
    }
    success: function(data) {

        // 请求成功，数据返回
    }
}, {

    progress: function(data) {

        // 正在读取，进度xxx
    }
})
```

```
// 动画模块，非常接近jQuery
$(‘header’).animate({
    width: ‘100px’,
    height: ‘100px’,
    left: 0,
    top: ‘100px’,
    opacity: 0.5

}, 2000)
```

```
// 路由模块
$route()
    .when([‘/’, ‘index’, ‘/index’], {

        template: ‘view/index.html’,
        js: ‘src/index.js’,
        css: ‘public/css/index.css’,
        callback: callback
    })
    .otherwise(‘/’)
    .scan()
```

```
// 数据模块
$(‘header’).data(‘name’, ‘头’)
```

```
// 选择器模块
var a = $('#a')
var b = $(window)
var c = $('#parent > div')
var d = $(a)
var e = $(document.createElement('span'))
var f = $(document)
var g = $(document.querySelectorAll('div'))
var h = $(divs)
var i = $('#a, #b, #parent')
var j = $('')
var k = $()
var l = $(null)
var m = $(undefined)
```

```
// 公共API
$.mix(a, b)        - 将对象b合并到对象a
$.create(‘div’)    - 创建一个div，返回一个fishbone对象
```

在高版本浏览器上，Shadow DOM是目前组件化的最佳解决方案。此外，React和Polymer是相对较好的方案
我们的组件化设计是二者的结合，最早的设计是参考Angular，但后来抛弃了那种复杂的写法
组件的两个版本设计demo都可以在Github的test下找到

参考：

- jQuery plugin
- Angular 1.x directive
- React JSX Component
- Polymer
- Shadow DOM

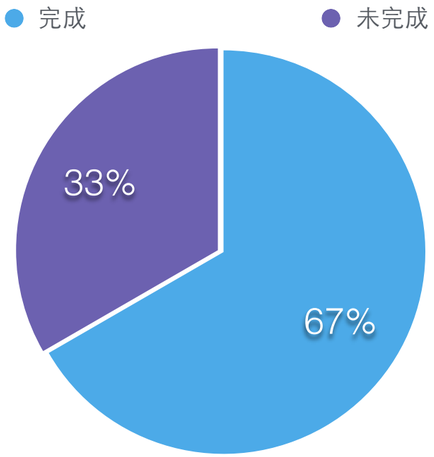
```
// 声明组件
var d = dropdown({
    selector: ‘dropdown’,
    id: ‘xx’,
    data: ‘hello, world’,
    view: ‘<div id=“{id”}>{data}</div>’,
    handleClick: function() {
        // clicked
    }
})
```

```
// 插入组件
<dropdown></dropdown>

// 自动替换
<div id=“xx”>hello,world</div>
```

```
// 修改数据
d.data = ‘hello, hiynn’
d.id = ‘hello’

// 自动替换
<div id=“hello”>hello,hiynn</div>
```

基础模块

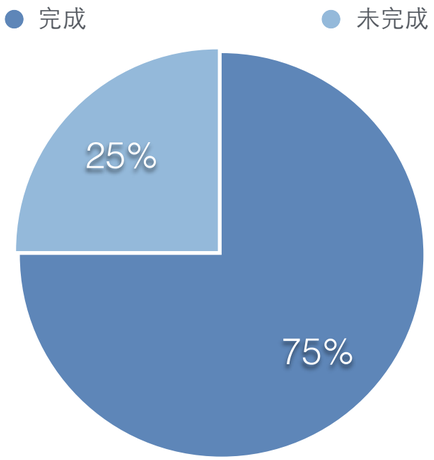
文件	描述	完成	行数
_sea-debug	Seajs	T	1125
_intro	头	T	11
main.js	种子模块	T	312
_fix	修复浏览器bug	F	8
prototype	原型扩展	T	98
lang	语言补全	F	9
data	数据缓存	F	45
_outro	尾	T	11
extend	对外接口	T	65



基础模块完成率

功能模块

文件	描述	完成	行数
http	请求，http、socket、	T	153
node	dom操作	T	345
event	事件	T	289
module	组件封装	F	73
css	css	T	132
attr	属性	T	198
route	路由	T	252
animate	动画	F	294



功能模块完成率

4. 图表组件 YCharts

优势

- 整合已有的图表
- 简单、风格一致的接口
- 可扩展，增加、删除图表和组件只修改配置文件
- 可重写，方便二次开发，满足定制化需求
- 可定制，开放图表的全部配置
- 规范代码，加强可读性

不足

- 没有开发完成
- 测试不充分
- 文档不完善
- 使用不够方便

Github：https://github.com/luankefei/chart-lib-master

5. CSS 预处理 Stylus

优势

- 可编程的CSS
- 类Jade的嵌套缩进语法
- 基于Node.js
- 易于上手

Github：https://github.com/stylus/stylus

```
// 简洁的语法
form input
    padding: 5px
    border: 1px solid
    border-radius: 5px
```

```
// 可编程
w1 = 100
w2 = 80

#test2
    background: blue
    height: (w1 - w2)px
```

```
// 基于node
// 安装
npm install -g stylus

// 执行
stylus index.styl -> index.css
```

七、工程化

1. Yeoman + Bower

- 自动生成
- 自动下载
- 结构统一

2. Gulp

- 项目打包
- 文件压缩
- 代码检查
- 自动刷新
- Http Server
- 自动测试
- 自动编译

3. Git

- 本地版本库
- 独立分支
- 离线提交
- 自动推送测试服务器

八、附录

1. 数据格式规范
2. 数据接口规范
3. JavaScript编码规范
4. Css编码规范
5. HTML编码规范
6. 文件命名规范

1. 数据格式规范

- 数据格式统一为JSON
- 返回的数据至少包含code（状态码）、和data（数据）两个字段。
- 返回的数据中不包括ASCII、Unicode等转义字符

```
{
    “code”: “0”,
    “data”: {
        “id”: “1”,
        “nickname”: “sunken”,
        “email”: “luankefei@hiynn.com”,
        “password”: “123456”,
        “group”: “admin”
    }
}
```

2. 数据接口规范

- 遵循RESTful风格
- 使用接口名描述资源
- 使用标准的HTTP方法

/user

/dashboard

/datasource

/datatable

八、XX

return;