Web前端设计文档

简述

此文档主要围绕网盘个人页面的架构设计来讨论，以下简称“网盘”，后台管理页面的实现原理相同

区别与其他的Web应用，网盘是一个富交互性、对交互、可扩展性、用户体验、方便定制开发等要求都很高的产品，需要依靠大量的Javascript代码来实现，此时设计一个灵活、方便扩展的架构就很有必要

情景分析

网盘的主要功能是围绕文件相关的操作，比如文件的列表、上传、移动、重命名等，可以抽象为有限的几个Model、Collection、View的操作，其中Model对应单个文件的属性，Collection对应一组Model的集合，View对应Model或Collection显示的模板。同时，对于文件的操作可能分散于多个不同的模块，比如Share模块赋予了文件分享的功能、Recycle模块提供了文件回收站的功能，Group模块提供了群组相关的操作，为了降低功能的耦合度，应该采用某种方式将各个功能模块的代码独立出来，并且抽象出可重用的方法或组件，避免不必要的重复代码。

设计模式

网盘设计为单页面Web应用（Single Page Application），所有的操作在同一个页面上进行，没有刷新，体验比较好，采用事件驱动模型解耦各模块代码，事件驱动是设计可扩展应用非常常用的一种手段，在Javascript中非常容易实现，方便添加第三方模块，扩展或改变原有的功能。对于复杂的数据交互部分采用自定义的jQuery UI组件XUI来实现，部分独立的界面扩展点，可采用局部MVC化。

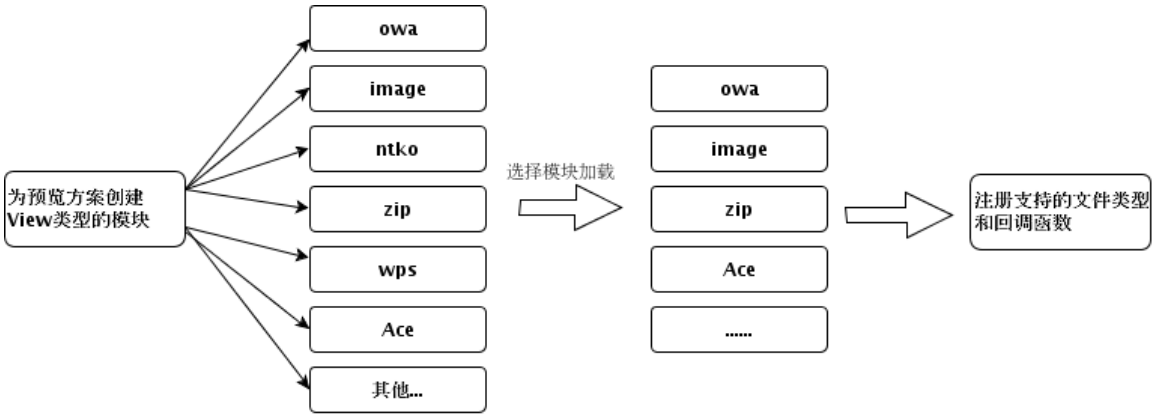
技术选型

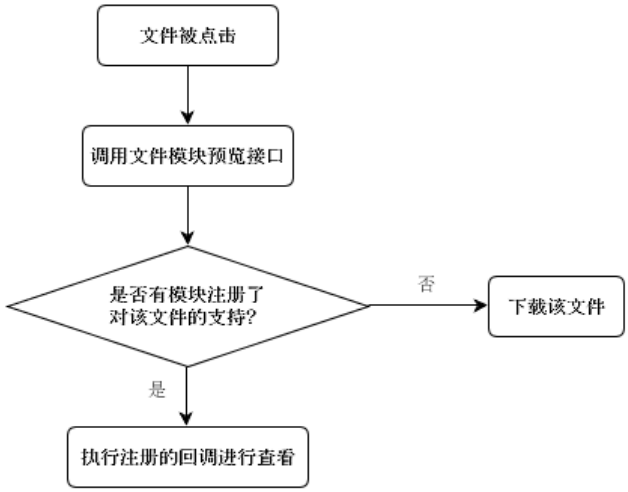
为了实现上述需求，网盘会用到但不限于以下的库或框架

1. SeaJS，SeaJS是一个遵循CommonJS规范的Javascript模块加载器，通过它可会很方便的将代码模块化，并且自动解决模块依赖问题，网盘使用它来进行模块的调度；
2. jQuery，jQuery是一个成熟的Dom和Ajax操作的库，网盘使用它来操作Dom和Ajax，屏蔽原生JS的跨浏览器问题；
3. XUI，XUI是针对网盘业务需求而专门开发的一套基于jQuery的UI组件库，包含了常用的UI组件，非常便于根据需求进行扩展，网盘的大部分数据交互都依赖于XUI来构建；
4. Underscore.js，Underscore是一个非常实用的Javascript库，它为操作Javascript对象、集合等提供了非常丰富的方法，网盘中有大量的对象集合操作，使用它可以简化非常多的代码，更见专注于业务的实现；
5. Backbone.js，Backbone为Web开发提供了Model、View、Collection、Router等功能，可以使Web应用数据与界面分离，更好的组织代码结构，网盘主要使用它来进行路由分发、历史记录、局部View的数据绑定等，便于提供更强大的扩展能力；

实现难点

1. 不同文件类型在不同场景下的处理流程
   当一个文档被点击时，需要根据不同的场景以及文档类型来判断采用哪种流程来处理，常规的实现方式是为每一种支持的文件类型都加一个分支判断，但这种方式带来的问题是不便于维护，也不利于扩展，比如某一天新增了某一种类型文档的查看支持，常规的方式需要改变既有的逻辑流程，形成越来越长的分支判断，难以维护，逻辑混乱，如果采用不同的方案，所支持的文档类型也可能会有差别，这里需要一种灵活方式来切换，具体设计实现如下：fscore作为文件操作核心模块，提供文件下载、上传、预览等基础功能，该模块提供了regView和fileView两个接口（fscore还提供了很多其他的文件操作相关的接口），其中regView用来给某些文档类型注册预览方案，每一种文件预览方案放到view目录下的一个独立模块里，比如owa、image、zip、ntko等等，通过Web端配置文件决定加载哪几个模块，当模块被加载时调用fscore模块的regView接口，注册该模块所支持的文档类型，并提供回调函数，当文档被点击时会调用fscore模块的fileView接口，该接口会根据当前文档的类型，去遍历已经注册了的文档类型，如果在注册列表里面找到了对该类型文件的支持，则执行对应的回调函数来进行预览，如果多个模块对同一种类型的文件行了注册，后注册的会覆盖之前注册的，如果遍历之后没有找到有模块对该类型文件进行注册，则调用下载接口下载该文件。可以看出，通过这种方式可以很方便的对文档的预览方案进行维护和切换，而不必在主流程里进行冗长的逻辑判断，如果后期增加了对某种文件的支持，则只需新增一个view模块，在其中注册对应的文件类型，然后通过配置文件将其加载即可，如果要改变某种文件默认的预览流程，也只需要在自定义的view模块里面将其覆盖就可以实现，整个流程大致如下：





1. 随文件目录和文件类型的变化决定按钮和菜单的显示
   不同的模块可能会给文件操作添加多个菜单或按钮，这些按钮会随着目录的切换和所选中文件的类型来决定是否显示或可用，这里采用的方式是，每个模块在注册文件操作菜单或按钮时，同时指定一个初始化判断函数，当触发文件右键、文件选中或者目录切换事件时，会遍历当前注册的菜单或按钮，将当前选中文件的Model作为参数传给判断函数，当函数返回为true时，才会显示该菜单或按钮，否则不予显示。

模块化

网盘每一部分独立的功能都放在一个独立的模块文件里面，通过SeaJS来进行管理和加载。

模块可以做的事：

a. 监听其他模块发出的事件；
b. 为其他模块添加功能，如预览方案的添加；
c. 触发自定义事件，便于其他模块进行监听；
d. 独立的模块功能；
e. 对外暴露接口，供其他模块调用；
f. 监听路由，提供路由映射到该模块的能力

打包时会自动将自定义模块合并压缩并且混淆代码

插件化扩展

如果想通过插件扩展或者改变现有的功能流程，可将这部分插件代码写在一个独立的模块里面，然后修改配置文件选择加载该模块即可，默认的工作流程也是按照这种方式来组织的

a. 支持哪几种类型的扩展？
   目前实现的扩展方式有：改变原有的功能流程，扩展原有的功能流程，为其他模块添加功能。
b. 如何实现？
   模块通过事件监听的方式来改变或者扩展原有的功能流程，通过调用其他模块提供的接口来添加功能。网盘中有两种事件触发机制

，使用由jQuery提供的trigger和triggerHandler方法，入口模块会注册一个全局事件对象AppEvent，其他所有模块均通过该对象注册和触发自定义事件，如果想为某个功能流程提供扩展点，则只需要在对应的位置触发一个自定义事件，事件的命名规则为“模块名.事件名"，其他模块在加载时监听该事件即可。如果触发事件时使用的是trigger，则会异步执行注册的回调列表，没有返回值，扩展功能流程，如果使用的是triggerHandler，则会执行最后一个注册的回调，并且获得回调的返回值，根据该返回值来决定下一步的动作，通过这种方式来改变原有的功能流程。有的模块可能会接受其他模块向其添加功能，比如前面讨论的多种预览方案的问题，这里采用的方式是该模块提供一个注册接口，其他模块如果想要为该模块添加功能，则在加载时调用该接口进行功能的添加，该模块在执行到相关位置时，会根据注册的功能列表来决定采用什么方式来处理。

工作流程

基本的工作流程如下：请求→加载入口模块→获取需要加载的模块列表→加载模块列表→模块注册路由监听、事件监听、向其他模块添加功能→执行各模块初始化方法→启动路由监听并执行分发→监听了该路由的执行方法→执行结果