angular 项目架构说明和设计指南

- Author: Liangdi(wu@liangdi.me)

- Last Updated (2017-07-12)

前置知识

- angular 框架 (https://angular.io)
- 遵循 angular 代码风格 (https://angular.io/guide/styleguide)
- 小写 `service` 为 angular 的 service , 大写 `Service` 为业务逻辑层 (Service)

基本架构

项目分为四层架构分别为：

- 视图层 (View):对应 angular 的 Template ,展示数据，完成和用户交互功能
- 控制器层 (Controller): 对应 angular 的 @Component 组件，主要是和 View 层实现 Mvvm 机制以及处理用户交互相关的业务逻辑
- 业务逻辑层 (Service): 以 angular 的 service 形式实现，注入到 Controller 中或者其他 Service 中，实现业务模块相关的业务逻辑
- 数据访问层 (Dao)：以 angular 的 service 形式实现, 注入到 Service 中，实现前端数据获取与存储的功能，包括 mock 数据，本地存储，前后端的 ajax/websocket 交互等

模块划分

- 模块划分依托于业务功能的层次,可以结合后端模块划分，进行匹配
- 通用模块使用 shared 命名
- 可以划分子模块，原则上不划分第三层，模块内部通用模块同样使用 shared 命名，仅内部使用
- 模块划分需要考虑路由配置，子模块使用子路由

组件封装

- 根据产品规格和原型/设计图梳理通用业务相关组件和通用业务功能，形成独立的组件或者模块，仅供本应用使用
- 业务无关组件和功能，形成独立组件和模块，可供全公司使用

View 层设计指南

- 组件需要明确定义 @Input 和 @Output
- 需要做一些业务功能无关的数据处理，使用 Pipe 实现

Controller 层设计指南

- Controller 中数据对象成员变量，需要明确到字段不得用 `field = {};` 的形式
- View 中需要访问的 service 限定为 public , 不需要访问的 service 限定为 private
- 使用组件封装复杂 View 层的业务功能，防止单个 Controller 代码量过大

Service 层设计指南

- 善用组合，避免代码量过大

Dao 层设计指南

- 需要设计接口，实现不同开发阶段对不同实现的可配置性，以及不同储存方案的可配置性
- Dao 层只注入到 Service 层中
- 在 @NgModule 中 providers 使用 provide(Service, {useClass: ServiceImpl} 配置
- 全局统一的数据处理，请求头处理，cookie处理等，在 Dao 层中实现， 模块业务相关的 统一数据处理在 Service 层中实现 ， 数据处理业务功能也设计为一个 angular 的 service

前后端数据格式

- 数据格式
- Http Method
- 提交参数说明