目录规划

主要目录结构

- `/commons` 存放公共的包，根据项目需要进行调整
- `/apps` 用于存放应用
- `/microservices` 用于存放微服务
- `/clouds` 用于存放云解决方案中的基础网元

```
/nev
    |- /commons
        |- /ubattery-base ;基础包
        |- /ubattery-utils  ;项目中使用到的工具包
        |- /ubattery-starter-security ; 安全相关的自动配置
        |- /ubattery-starter-swagger2 ;自动配置 swagger2 文档，引入依赖即可使用
    |- /apps
        |- /ubattery-nev ; 新能源汽车全生命周期监管子系统
        |- /ubattery-chargingfacility ;  新能源汽车充电设施全生命周期监管子系统
        |- /ubattery-evcell ; 新能源汽车动力电池全生命周期监管子系统
        |- /ubattery-hydrogen ; 加氢站全生命周期监管子系统
        |- /ubattery-v2x ; 智能网联汽车管理子系统
    |- /microservices ; 面向应用的服务
        |- /ubattery-user-service ; 用户管理
        |- /ubattery-organization-service ; 组织管理
        |- /ubattery-nev-service ; 车辆服务
        |- /ubattery-chargingfacility-service ; 充电设施服务
        |- /ubattery-evcell-service ;  动力电池服务
        |- /ubattery-RealtimeAlarm-service ; 实时告警服务
        |- /ubattery-FailurePrediction-service; 故障预测模型服务
        |- /ubattery-MetrologySupport-service; 计量保障服务
        |- /ubattery-v2x-service; 智能网联汽车服务
    |- /clouds
        |- /ubattery-access-gateway ; 接入网关
        |- /ubattery-oauth2-server ; 安全认证服务
        |- /ubattery-gateway-server ; 网关
    |- /deps ;存放项目依赖的第三方组件（无法在maven中找到），工具等
    |- /env ; 用于存放开发工具的通用配置等
    |- /tests ; 存放测试相关的配置，脚本，环境，工具等
    |- /production ; 用于存放生产环境相关的配置
```

通用目录结构

应用、服务、组件等都包含以下目录结构

```
|- /<path>/
  |- /docs/   ; 存放文档，建议使用 Markdown 格式编写，插图等文件存放到 assets/ 目录下
    |- /assets/ ; 插图等相关资源
  |- /docker/ ; 存放构建 docker 环境的相关脚本
  ... ;项目其它文件
```

举例说明:

```
/nev/
    |- /microservices
        |- /ubattery-user-services
            |- /docs
                |- /assets
            |- /docker
            |- /src
                |- /main
                    |- /java
                    |- /resources
                |- /test
                    |- /java
                    |- /resources
```

基础概念

应用

- 应用是面向业务，为业务人员提供交互界面的软件包。按类型分，可以分为Web应用和移动应用。
- 应用采用前后端分离，前端负责交互，后端负责数据的管理、业务逻辑的处理、外部服务的调用等

微服务

- 微服务是能独立部署的应用
- 微服务包含服务端口(对于web服务来说，就是 restful 的接口)，服务运行需要的资源，如：MySQL，Hadoop等，这些资源可以被看做服务独立占用的资源，在服务性能受到影响的时候，可以通过扩展资源来提供更好的服务
- 微服务的调用需要考虑安全性，避免有害的调用影响服务
- 系统被拆成微服务以后，一个业务请求，就由多个分布式的服务来完成，这时进行跟踪排错就变得困难，需要引入log trace 的机制，和log分析工具来帮助分析
- 系统微服务化以后，服务发生故障无法访问的情况属于常见情况，这时要避免一个服务停止服务，造成整个系统的停机，通过熔断机制可以实现对服务的包含

云网元

- 组成解决方案的基础服务
- 安全认证服务，网关等为常见网元