**第1章 服务端**

**1.1 环境要求**

**基础开发环境**

* JDK: 1.8
* Maven: 3.3+
* MySql: 5.7+
* Redis: 4.0+
* Node Js: 10.15.0+
* Npm: 5.6.0+
* yarn:

**IDE插件**

* Lombok Plugin （必装）
* MybatisX Plugin （idea选装）

**推荐IDE**

**后端使用**

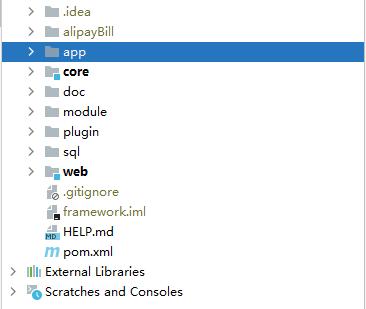
* IntelliJ IDEA

**前端使用**

* IntelliJ WebStorm

**1.2 工程导入**

若看到如下工程形态，说明启动成功



**1.2 工程配置**

1、进入Web工程下application.yml ，进行端口、数据库驱动类的配置

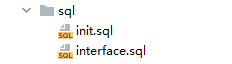


**1.2 建立数据库**

1、运行项目WebApplication,启动后自动创建数据表



2、执行sql文件夹下的初始化脚本



**1.3 数据库结构**

**Table: sys\_dept（机构表）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| dept\_code | varchar |  |  |  | 编码 |
| dept\_name | varchar |  |  |  | 名称 |
| dept\_type | varchar |  |  |  | 类型 |
| parent\_code | varchar |  |  |  | 父编码 |
| parent\_codes | varchar |  |  |  | 所有父编码 |

**Table: sys\_config（系统参数）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| config\_key | varchar |  |  |  | 参数KEY |
| config\_name | varchar |  |  |  | 参数名称 |
| config\_type | varchar |  |  |  | 参数类型 |
| config\_value | varchar |  |  |  | 参数值 |
| remarks | varchar |  |  |  | 备注 |

**Table: sys\_dict\_data（系统字典数据）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| dict\_label | varchar |  |  |  | 字典标签 |
| dict\_type | varchar |  |  |  | 字典类型 |
| dict\_value | varchar |  |  |  | 字典值 |

**Table: sys\_dict\_data（系统字典类型）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| dict\_name | varchar |  |  |  | 字典名称 |
| dict\_type | varchar |  |  |  | 字典类型 |
| remark | varchar |  |  |  | 备注 |

**Table: sys\_dict\_data（系统字典类型）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| dict\_name | varchar |  |  |  | 字典名称 |
| dict\_type | varchar |  |  |  | 字典类型 |
| remark | varchar |  |  |  | 备注 |

**Table: sys\_role（角色表）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| code | varchar |  | UK |  | 角色编码 |
| name | varchar |  |  |  | 角色名称 |
| remark | varchar |  |  |  | 备注 |

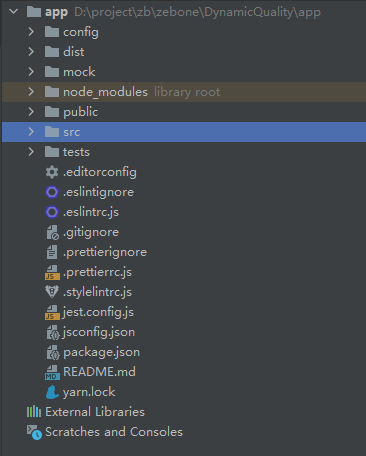
**Table: sys\_user（用户表）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Remarks** |
| id | varchar |  | PRI |  | 主键 |
| username | varchar |  | UK |  | 账号 |
| password | varchar |  |  |  | 密码 |
| name | varchar |  |  |  | 姓名 |
| roles | varchar |  |  |  | 角色编码集合 |

**第2章 客户端**

**2.1 工程导入**

导入项目app文件夹的工程



**2.2 安装第三方依赖**

当前根目录下进入控制台，输入yarn install

**2.3 设置代理**

在config/proxy.js中设置服务端代理



**2.4 启动运行**

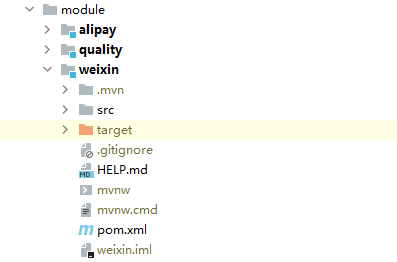
控制台输入：npm run start

**第3章 项目结构**

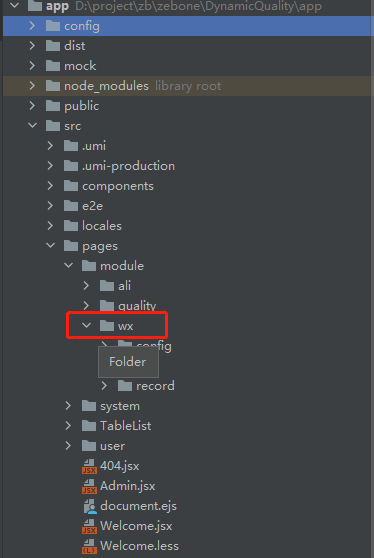
**3.1 微信项目**

服务端

module文件夹下的weixin项目



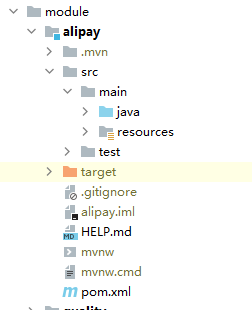
客户端



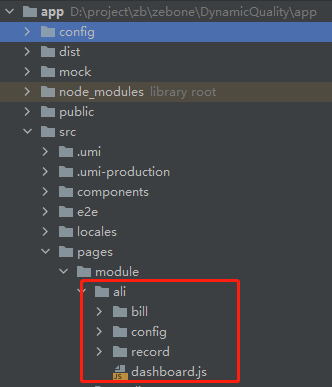
**3.2 支付宝项目**

服务端

module文件夹下的alipay项目



客户端



**第4章 开发步骤**

**4.1 标题和 Logo**

在实际使用中我们一般会通过 config\defaultSettings.ts 来控制标题和 Logo，这部分功能来自 [ProLayout](https://procomponents.ant.design/components/layout) 的功能。

const settings: LayoutSettings & {

pwa?: boolean;

logo?: string;

} = {

// 修改右上角的 logo

logo: 'https://gw.alipayobjects.com/zos/rmsportal/KDpgvguMpGfqaHPjicRK.svg',

// 设置标题的 title

title: 'Ant Design Pro',

navTheme: 'light',

// 拂晓蓝

primaryColor: '#1890ff',

layout: 'mix',

contentWidth: 'Fluid',

fixSiderbar: true,

};

export default settings;

如果你需要动态的修改标题或者 Logo，就需要使用运行时的能力了。我们可以在 src\app.tsx 中做如下配置：

// https://umijs.org/zh-CN/plugins/plugin-layout

export const layout: RunTimeLayoutConfig = ({ initialState }) => {};

其中的 initialState 就是通过初始化插件获得数据，每次 initialState 改变就会触发 layout 的重新渲染，我们就可以根据 initialState 来自定义 title，看起来像是这样的，Logo 也是同理。

// https://umijs.org/zh-CN/plugins/plugin-layout

export const layout: RunTimeLayoutConfig = ({ initialState }) => {

return {

title: initialState.serverName,

};

};

同时 ProLayout 会根据菜单和路径来自动匹配浏览器的标题。欢迎页就会显示 欢迎 - Ant Design Pro ，如果你不喜欢这样可以设置 pageTitleRender=false 来关掉它。

如果需要根据内容动态更新页面标题，则可以使用浏览器 document.title API。 对于更复杂的场景，当您想从 React 组件更改标题时，可以使用第三方库 React Helmet。

**4.2 favicon**

favicon 是展示在浏览器标签页上的内容，严格来说它是属于浏览器 meta 的一部分，浏览器认为 favicon 不会经常改动做了非常强的缓存。所以我们并没有做动态修改 favicon 的方案。

Pro 的默认 favicon 的配置存在于 src\pages\document.ejs, 我们可以在这里写 html 相关的代码, 我们默认配置了 public 文件夹中的 favicon.ico 文件，你可以直接覆盖来修改。当然也可以修改 href 为你想要的 cdn 地址。

<link rel="icon" href="<%= context.config.publicPath +'favicon.ico'%>" type="image/x-icon" />

**4.3 加载页**

**js 加载前**

首先是的在 js 还没加载成功，但是 html 已经加载成功的 landing 页面。这个页面的配置存在于 src\pages\document.ejs 中。它使用了 home\_bg.png ,pro\_icon.svg 和 KDpgvguMpGfqaHPjicRK.svg 三个带有品牌信息的图片，你可以按需修改他们。

**js 加载后**

如果我们在项目中打开了[代码分割](https://umijs.org/zh-CN/config#dynamicimport)的话，在每次路由切换的时候都会进入一个加载页面。

dynamicImport: {

loading: '@ant-design/pro-layout/es/PageLoading',

}

**业务中的加载**

在实际的项目中，我们需要等待用户信息或者鉴权系统的请求完成后才能展示页面。所以我们让 getInitialState支持了异步请求，同时在请求时会停止页面的渲染。这种情况下也是需要一个加载页的。我们可以在 src\app.tsx 中配置：

/\*\* 获取用户信息比较慢的时候会展示一个 loading \*/

export const initialStateConfig = {

loading: <PageLoading />,

};

**4.3 新增页面**

在 src / pages 下创建新的 js，less 文件。 如果有多个相关页面，您可以创建一个新文件夹来放置相关文件。

config

src

models

pages

+ NewPage.js

+ NewPage.less

...

...

package.json

为了更好的演示，我们初始化NewPage.js的内容如下：

export default () => {

return <div>New Page</div>;

};

**4.4 将文件加入菜单和路由**

我们需要在 routes.ts 中使用 component 配置我们页面到路由中。

export default [

{

path: '/user',

component: '../layouts/UserLayout',

routes: [

{

// path 支持为一个 url，必须要以 http 开头

path: 'https://pro.ant.design/docs/getting-started-cn',

target: '\_blank', // 点击新窗口打开

name: '文档',

},

{

// 访问路由，以 / 开头为绝对路径

path: '/user/login',

// ./Page ->src/pages/Login

component: './NewPage',

},

{

// 访问路由，如果不是以 / 开头会拼接父路由

// reg -> /user/reg

path: 'reg',

// ./Page ->src/pages/Reg

component: '../layouts/NewPage2',

},

],

},

];

路由配置完成后，访问页面即可看到效果，如果需要在菜单中显示，需要配置 name，icon，hideChildrenInMenu等来辅助生成菜单。

具体值如下：

* name:string 配置菜单的 name，如果配置了国际化，name 为国际化的 key。
* icon:string 配置菜单的图表，默认使用 antd 的 icon 名，默认不适用二级菜单的 icon。
* access:string 权限配置，需要预先配置权限
* hideChildrenInMenu:true 用于隐藏不需要在菜单中展示的子路由。
* hideInMenu:true 可以在菜单中不展示这个路由，包括子路由。
* hideInBreadcrumb:true 可以在面包屑中不展示这个路由，包括子路由。
* headerRender:false 当前路由不展示顶栏
* footerRender:false 当前路由不展示页脚
* menuRender: false 当前路由不展示菜单
* menuHeaderRender: false 当前路由不展示菜单顶栏
* flatMenu 子项往上提，只是不展示父菜单

**第5章 部署安装**

**5.1 服务端**

项目根目录下执行mvn clean package

**5.2 客户端**

项目根目录下执行npm run build