---

link: null
title: 珠峰架构师成长计划
description: 安装依赖
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=137 sentences=289, words=1506

1.前端项目最佳实践

1. 工具选择

类别 选择 框架
react (https://reactjs.org/)

JS语言
TypeScript (http://www.typescriptlang.org)

CSS语言
css-modules (https://github.com/css-modules/css-modules) less (http://lesscss.org) postcss (https://github.com/postcss/postcss)

JS编译
babel (https://www.babeljs.cn/)

模块打包
webpack全家桶 (https://webpack.github.io/)

单元测试
jest (https://github.com/facebook/jest) enzyme (https://github.com/airbnb/enzyme) puppteer (https://github.com/puppeteer/puppeteer) jsdom (https://github.com/jsdom/jsdom)

路由
react-router (https://github.com/ReactTraining/react-router)

数据流
dva (https://dvajs.com) redux生态 (https://www.redux.org.cn/)

代码风格
eslint (https://eslint.org/) prettier (https://prettier.io/)

JS压缩
TerserJS (https://github.com/terser/terser)

CSS压缩
cssnano (https://github.com/cssnano/cssnano)

请求库
umi-request (https://github.com/umijs/umi-request#readme)

UI
AntDesign (https://ant.design/docs/react/introduce-cn) AntDesignPro (https://pro.ant.design/index-cn)

国际化
react-intl (https://github.com/formatjs/react-intl)

hooks库
umi-hooks (https://hooks.umijs.org/)

静态文档
docz (https://www.docz.site/)

微前端
qiankun (https://github.com/umijs/qiankun)

图表库
antv (https://antv.vision/)

2.技术栈选型

2.1 固定化

- React框架
- TypeScript语言
- Less+CSS Modules
- Eslint+Prettier+固定配置
- 固定数据流方案dva
- 固定babel插件
- Jest+Enzyme
- 框架版本不允许锁定，^前缀必须有
- 主要依赖不允许自定义依赖版本

2.2 配置化

- 不仅是框架功能，还有UI界面
- 路由、布局、菜单、导航、面包屑、权限、请求、埋点、错误处理
- 只管写Page页面就可以了

2.2.1 编译态配置

- 给node.js使用，比如webpack、babel相关配置，静态路由配置

2.2.2 运行态配置

- 给浏览器用、比如渲染逻辑、动态修改路由、获取用户信息

2.3 约定化

- 国际化
- 数据流
- MOCK
- 目录结构
- 404
- 权限策略
- Service
- 配置文件

1.3 理念

- 通过最佳实践减少不必要的选择的差异
- 通过插件和插件集的架构方式，满足不同场景的业务
- 通过资产市场和场景市场着力解决70%的开发者问题
- 通过对垂直场景采取强约束的方式，进一步提升研发效率
- 不给选择、配置化、约定化

2.前端

2.1. Ant Design Pro项目初始化

- pro.ant.design (https://pro.ant.design/docs/getting-started-cn)
- Pro的区块 (https://github.com/ant-design/pro-blocks)
- ant-design-pro-layout (https://github.com/ant-design/ant-design-pro-layout)
- ant-design-pro-layout (https://ant-design.github.io/ant-design-pro-layout/?path=/story/basiclayout--readecn)
- pro-blocks (https://github.com/ant-design/pro-blocks)
- uset-typescript-cn (https://pro.ant.design/docs/uset-typescript-cn)
- umi-config (https://umijs.org/zh/config/#chainwebpack)
- Ant Design Pro 是一个企业级中后台前端/设计解决方案，我们秉承 Ant Design 的设计价值观，致力于在设计规范和基础组件的基础上，继续向上构建，提炼出典型模板/业务组件/配套设计资源，进一步提升企业级中后台产品设计研发过程中的『用户』和『设计者』的体验。

2.2 启动项目

2.2.1 安装

- 新建一个空的文件夹作为项目目录，并在目录下执行
- python-380 (https://www.python.org/downloads/release/python-380/)

```
npm config set python "C:/Python38/python.exe"
yarn create umi
```

2.2.2 目录结构

- 我们已经为你生成了一个完整的开发框架，提供了涵盖中后台开发的各类功能和坑位，下面是整个项目的目录结构。

```
├── config                  # umi 配置，包含路由，构建等配置
├── mock                    # 本地模拟数据
├── public
│   └── favicon.png         # Favicon
├── src
│   ├── assets              # 本地静态资源
│   ├── components          # 业务通用组件
│   ├── e2e                 # 集成测试用例
│   ├── layouts             # 通用布局
│   ├── models              # 全局 dva model
│   ├── pages               # 业务页面入口和常用模板
│   ├── services            # 后台接口服务
│   ├── utils               # 工具库
│   ├── locales             # 国际化资源
│   ├── global.less         # 全局样式
│   └── global.ts           # 全局 JS
├── tests                   # 测试工具
├── README.md
└── package.json
```

2.2.3 本地开发

安装依赖

```
git init
npm install
```

启动项目

```
npm start
```

2.3 用户注册

- umijs-block (https://umijs.org/zh/guide/block.html)
- pro区块 (https://pro.ant.design/docs/block-cn)
- pro-dashboard (https://preview.pro.ant.design/dashboard/analysis)
- pro-blocks下载 (https://github.com/ant-design/pro-blocks)

2.3.1 先配置区块下载地址

config.ts

```
export default {
  plugins,
+ block: {
+   defaultGitUrl: 'https://github.com/ant-design/pro-blocks',
+ },
```

2.3.2 区块列表

```
umi block list
```

```
UserRegister (https:

UserRegisterResult  (https:
请输入输出安装区块的路径 /user/register-result
```

2.3.3 user\register\index.tsx

src\pages\user\register\index.tsx

```
  onGetCaptcha = () => {
+    const { dispatch, form } = this.props;
+    const mobile = form.getFieldValue('mobile');
+    dispatch({
+      type: 'login/getCaptcha',
+      payload: mobile,
+    })
    let count = 59;
    this.setState({ count });
    this.interval = window.setInterval(() => {
      count -= 1;
      this.setState({ count });
      if (count
        clearInterval(this.interval);
      }
    }, 1000);
  };

+
+            {getFieldDecorator('currentAuthority', {
+              rules: [
+                {
+                  required: true,
+                  message: formatMessage({ id: 'userandregister.currentAuthority.required' }),
+                }
+              ],
+            })(
+
+                普通用户
+                管理员
+
+            )}
+
```

2.3.4 user\register\locales\zh-CN.ts

src\pages\user\register\locales\zh-CN.ts

```
+  'userandregister.currentAuthority.placeholder': '角色',
+  'userandregister.currentAuthority.required': '请输入邮箱地址！',
```

2.3.5 user\register\service.ts

src\pages\user\register\service.ts

```
import request from '@/utils/request';
import { UserRegisterParams } from './index';

export async function fakeRegister(params: UserRegisterParams) {
+  return request('/server/api/register', {
    method: 'POST',
    data: params,
  });
}
```

2.3.6 src\services\login.ts

src\services\login.ts

```
export async function getFakeCaptcha(mobile: string) {
+  return request(`/server/api/login/captcha?mobile=${mobile}`);
}
```

2.4. 用户登录

2.4.1 services\login.ts

src\services\login.ts

```
export async function fakeAccountLogin(params: LoginParamsType) {
+  return request('/server/api/login/account', {
    method: 'POST',
    data: params,
  });
}
```

2.4.2 src\services\user.ts

src\services\user.ts

```
export async function queryCurrent(): Promise {
+  return request('/server/api/currentUser');
}
```

2.5. 权限菜单

2.5.1 src\models\login.ts

src\models\login.ts

```
if (response.status
+        if (response.token) {
+          localStorage.setItem('token', response.token);
+        }
        const urlParams = new URL(window.location.href);
```

2.5.2 src\utils\request.ts

src\utils\request.ts

```
const request = extend({
  errorHandler, // 默认错误处理
  credentials: 'include', // 默认请求是否带上cookie
});
+const baseURL = 'http://localhost:4000';
+request.interceptors.request.use((url: any, options: any) => {
+  if (localStorage.getItem('token')) {
+    options.headers.Authorization = 'Bearer ' + localStorage.getItem('token')
+  }
+  if (url.startsWith('/server')) {
+    url = baseURL + url.slice(7);
+  }
+  return { url, options };
+});

export default request;
```

2.6 docker

2.6.1 Dockerfile

```
FROM nginx
LABEL name="antdesign-front"
LABEL version="1.0"
COPY  ./dist/ /usr/share/nginx/html/
COPY ./antdesign-front.conf /etc/nginx/conf.d/
EXPOSE 80
```

2.6.2 .dockerignore

```
.git
node_modules
package-lock.json
Dockerfile
.dockerignore
```

2.6.3 antdesign-front.conf

antdesign-front.conf

```
server {
    listen       80;
    server_name  47.104.204.74;
    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
        try_files $uri $uri/ /index.html;
    }
}
```

3.后端

3.1 api.js

```
let express = require("express");
let bodyParser = require("body-parser");
let jwt = require('jwt-simple');
let cors = require("cors");
let Models = require('./db');
let sendCode = require('./sms');
let session = require("express-session");
let MongoStore = require('connect-mongo')(session);
let config = process.env.NODE_ENV == 'production' ? require('./config/config.prod') : require('./config/config.dev');
let app = express();
app.use(
    cors({
        origin: config.origin,
        credentials: true,
        allowedHeaders: "Content-Type,Authorization",
        methods: "GET,HEAD,PUT,PATCH,POST,DELETE,OPTIONS"
    })
);
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());
app.use(
    session({
        secret: config.secret,
        resave: false,
        saveUninitialized: true,
        store: new MongoStore({
            url: config.dbUrl,
            mongoOptions: {
                useNewUrlParser: true,
                useUnifiedTopology: true
            }
        })
    })
);
app.get('/', async (req, res) => {
    res.json({ code: 0, data: `hello` });
});
app.get('/api/login/captcha', async (req, res) => {
    let mobile = req.query.mobile;
    let captcha = rand();
    req.session.captcha = captcha;
    await sendCode(mobile, captcha);
    res.json({ code: 0, data: `[仅限测试环境验证码]: ${captcha}` });
});
app.post('/api/register', async (req, res) => {
    let user = req.body;
    if (user.captcha != req.session.captcha) {
        return res.json({ code: 1, error: '验证码不正确' });
    }
```

```
    let avatarValue = require('crypto').createHash('md5').update(user.mail).digest('hex');
    user.avatar = `https://secure.gravatar.com/avatar/${avatarValue}?s=48`;
    user = await Models.UserModel.create(user);
    res.send({ status: 'ok', currentAuthority: 'user' });
});
app.post('/api/login/account', async (req, res) => {
    let user = req.body;
    let query = {};
    if (user.type == 'account') {
        query.mail = user.userName;
    } else if (user.type == 'mobile') {
        query.mobile = user.mobile;
        if (user.captcha != req.session.captcha) {
            return res.send({
                status: 'error',
                type: user.type,
                currentAuthority: 'guest',
            });
        }
    }
    let dbUser = await Models.UserModel.findOne(query);
    if (dbUser) {
        dbUser.userid = dbUser._id;
        dbUser.name = dbUser.mail;
        let token = jwt.encode(dbUser, config.secret);
        res.send({ status: 'ok', token, type: user.type, currentAuthority: dbUser.currentAuthority });
    } else {
        return res.send({
            status: 'error',
            type: user.type,
            currentAuthority: 'guest',
        });
    }
});

app.get('/api/currentUser', async (req, res) => {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            let user = jwt.decode(authorization.split(' ')[1], config.secret);
            user.userid = user._id;
            user.name = user.mail;
            res.json(user);
        } catch (err) {
            res.status(401).send({});
        }
    } else {
        res.status(401).send({});
    }

});
app.listen(4000, () => {
    console.log('服务器在4000端口启动!');
});

function rand() {
    let min = 1000, max = 9999;
    return Math.floor(Math.random() * (max - min)) + min;
}
```

3.2 db.js

db.js

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const ObjectId = Schema.Types.ObjectId;
let configs = process.env.NODE_ENV == 'production' ? require('./config/config.prod') : require('./config/config.dev');
const conn = mongoose.createConnection(configs.dbUrl, { useNewUrlParser: true, useUnifiedTopology: true });
const UserModel = conn.model('User', new Schema({
    userid: { type: String },
    name: { type: String },
    mail: { type: String, required: true },
    password: { type: String, required: true },
    mobile: { type: String, required: true },
    avatar: { type: String, required: true },
    currentAuthority: { type: String, required: true }
}));

module.exports = {
    UserModel
}
```

3.3 config.prod.js

config\config.prod.js

```
module.exports = {
    secret: 'zhufengcms',
    dbUrl: "mongodb://antdesign-mongo:27017/zhufengcms",
    origin: ["http://47.104.204.74", "http://47.104.204.74:8000"]
}
```

3.4 config\config.dev.js

config\config.dev.js

```
module.exports = {
    secret: 'zhufengcms',
    dbUrl: "mongodb://localhost:27017/zhufengcms",
    origin: ["http://localhost:8000"]
}
```

3.5 sms.js

sms.js

```
const axios = require('axios');
const smsConfig = require('./smsConfig');

module.exports = async (mobile, captcha) => {
    const url = 'https://open.ucpaas.com/ol/sms/sendsms';
    let result = await axios({
        method: 'POST',
        url,
        data: {
            sid: smsConfig.sid,
            token: smsConfig.token,
            appid: smsConfig.appid,
            templateid: smsConfig.templateid,
            param: captcha,
            mobile
        },
        headers: {
            "Content-Type": "application/json;charset=utf-8",
            "Accept": "application/json"
        }
    })
    return result;
}
```

3.6 smsConfig.js

smsConfig.js

```
module.exports = {
    sid: '32548fb951ac0df279db0e6e9a515566',
    token: 'aa0309c08920ca38201de69eb3c745b6',
    appid: '16129d504b7c484c9e8f09b4ec929983',
    templateid: '387675'
}
```

3.7 package.json

package.json

```
  "scripts": {
    "start": "node api.js"
  },
```

3.8 Dockerfile

Dockerfile

```
FROM node
ENV NODE_ENV production
LABEL name="antdesign-server"
LABEL version="1.0"
COPY . /app
WORKDIR /app
RUN npm install
EXPOSE 4000
CMD npm start
```

3.9 .dockerignore

.dockerignore

```
.git
node_modules
package-lock.json
Dockerfile
.dockerignore
```

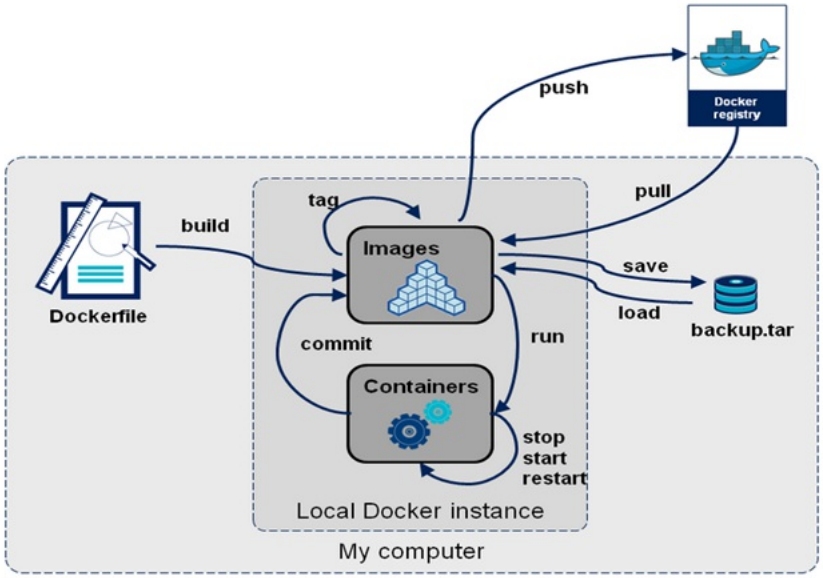
4.服务器布署

4.1 安装配置服务器

```
#升级所有包同时也升级软件和系统内核
yum update  -y
#只升级所有包，不升级软件和系统内核
yum upgrade
```

4.2 docker是什么？#

- Docker 属于 Linux 容器的一种封装，提供简单易用的容器使用接口。
- Docker 将应用程序与该程序的依赖，打包在一个文件里面。运行这个文件，就会生成一个虚拟容器。程序在这个虚拟容器里运行，就好像在真实的物理机上运行一样



4.3 安装docker

```
yum install -y yum-utils   device-mapper-persistent-data   lvm2
yum-config-manager \
    --add-repo \
    https:
yum install -y docker-ce docker-ce-cli containerd.io
```

4.4 阿里云加速

```
mkdir -p /etc/docker
tee /etc/docker/daemon.json <
```

4.5 安装git

```
yum install git -y
```

4.6 安装node和npm

- nvm (https://github.com/nvm-sh/nvm)

```
curl -o- https:

source /root/.bashrc
nvm ls-remote
nvm install v12.13.1
npm i cnpm -g
```

4.7 创建docker网络

```
docker network create --driver bridge antdesign
docker network connect  antdesign antdesign-mongo
docker network inspect antdesign
```

4.8 启动mongodb

4.8.1 启动mongo的docker容器

```
docker pull mongo:latest
docker images
docker run -d --name antdesign-mongo -p 27017:27017 --net antdesign mongo
docker ps
docker exec -it antdesign-mongo bash
mongo
```

4.8.2 本地安装mongodb

4.8.2.1 配置MongoDB的yum源

```
vim /etc/yum.repos.d/mongodb-org-4.0.repo
#添加以下内容：
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0/x86_64/
gpgcheck=0
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc

#这里可以修改 gpgcheck=0, 省去gpg验证
[root@localhost ~]# yum makecache
```

4.8.2.2 安装MongoDB

```
yum -y install mongodb-org
whereis mongod
vim /etc/mongod.conf
```

4.8.2.3 启动MongoDB

```
systemctl start mongod.service #启动
systemctl stop mongod.service #停止
systemctl status mongod.service #查看状态
```

4.8.2.4 外网访问

```
systemctl stop firewalld.service #停止firewall
systemctl disable firewalld.service #禁止firewall开机启动
```

4.9 启动后台服务器

```
git clone https:
cd antdesign-server
docker build -t antdesign-server .
docker image ls
docker run  -d --name antdesign-server -p 4000:4000 --net antdesign antdesign-server
curl http:
```

4.10 启动前台服务

```
git clone https:
cd antdesign-front
cnpm install
cnpm run build

docker build -t antdesign-front .
docker run -d --name antdesign-front -p 80:80  --net antdesign antdesign-front
```