人力资源后台项目

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

项目架构

底层框架

vue-element-admin

基础模版 | 目录结构 | 资源/图片 | git | 配置文件

业务模块

RBAC权限

业务模块

登录 | 主页 | 组织 | 角色 | 员工 | 权限

解决方案

左树右表 | 弹层编辑 | 行内编辑 | 跳转编辑 | 图表渲染 | 表单校验

组件封装 | Vuex应用 | 前端持久化 | 树形递归 | cos上传 | 联合查询

excel导出 | 权限应用 | 自定义指令 | 反向代理 | 主页鉴权 | nginx部署

课程安排

- 基础环境+登录（day01）
- 登录 + 主页鉴权（day02）
- 主页模块（day03）
- 组织架构（day04）
- 角色管理（day05）
- 左树右表（day06）
- 员工新增修改（day07）
- cos上传+权限数据(day08)
- 权限应用(day09)
- 首页 + 部署(day10)

学完该课程具备

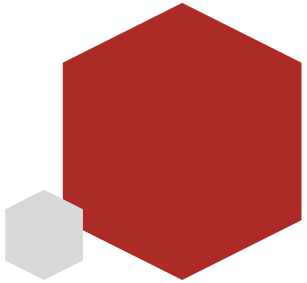
基于vue-element-admin二次开发	树形组件及树形数据处理	RBAC动态路由
前端项目CRUD(增删改查)	自定义指令解决权限技术方案	echarts图表应用
封装组件适配产品需求	Cos第三方上传	请求工具封装
三种编辑模式(弹层，编辑，跳转)	左树右表-联合查询	主页鉴权
反向代理解决跨域	封装请求工具的项目应用	nginx项目部署上线
表单验证及自定义校验	Vuex技术在项目中的应用	excel的导入导出

课程资料

vue-element-admin文档地址：https://panjiachen.gitee.io/vue-element-admin-site/zh/

vue-element-admin演示地址：https://panjiachen.github.io/vue-element-admin/

项目演示地址：https://heimahr.itheima.net



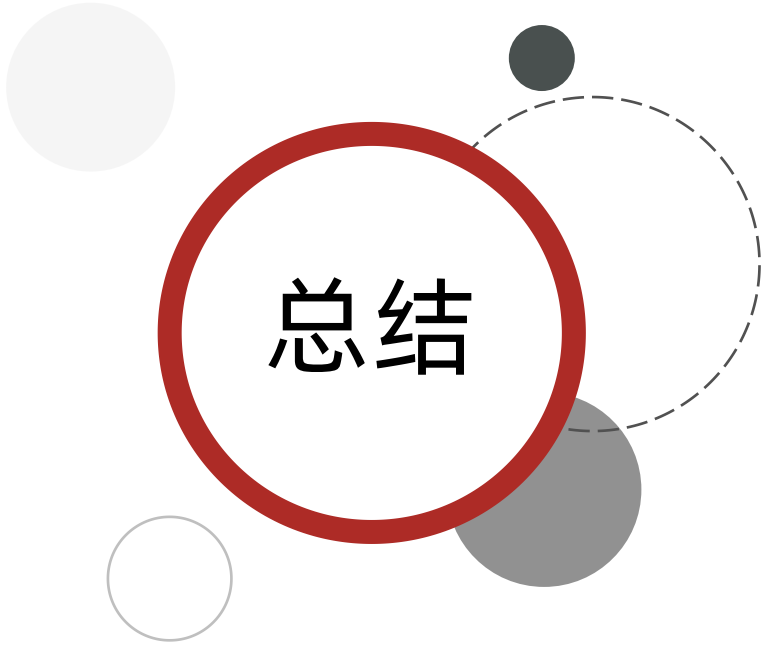
拉取项目基础代码

拉取模版代码

git clone https://github.com/PanJiaChen/vue-admin-template.git heimahr

其他：core-js版本处理

- 项目模版中的core-js的版本号有些滞后，需要将其版本号改为“3.25.5”再安装依赖



1. 使用模版是更快更好更贴近企业的开发模式

```
https://github.com/PanJiaChen/vue-admin-template.git
```

2. 修改core-js版本为3.25.5并安装依赖

```
core-js: 3.6.5 ➡  3.25.5
```

```
yarn
```

3. 启动命令在package.json中查看

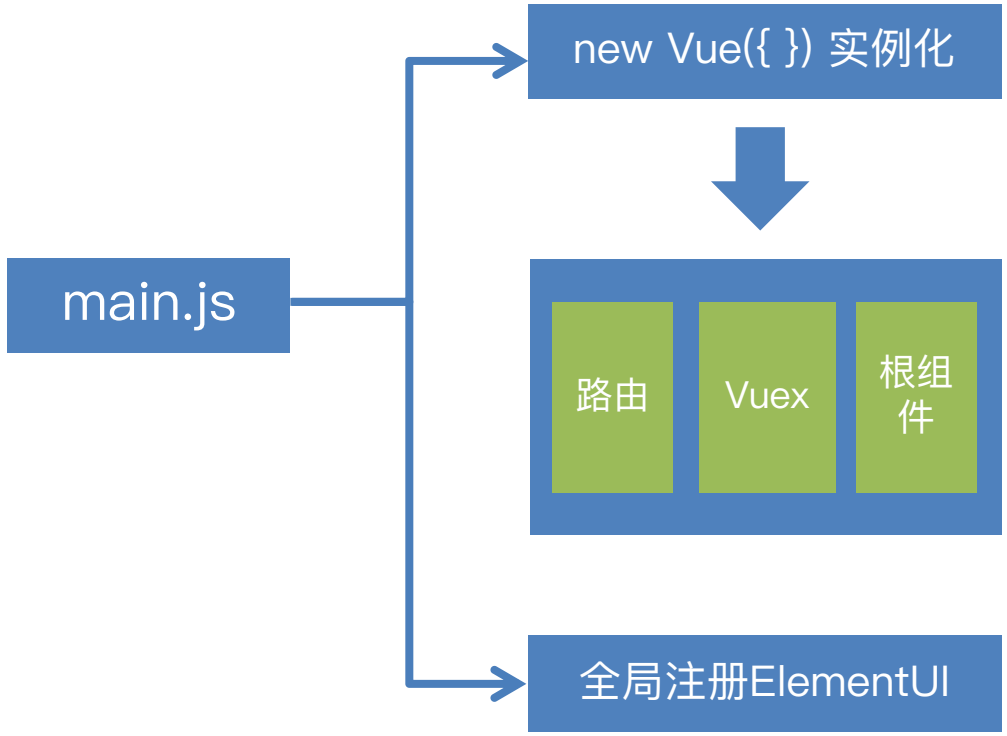
```
yarn dev
```

4. 插件是提升开发效率的利器：ESLint & Vetur

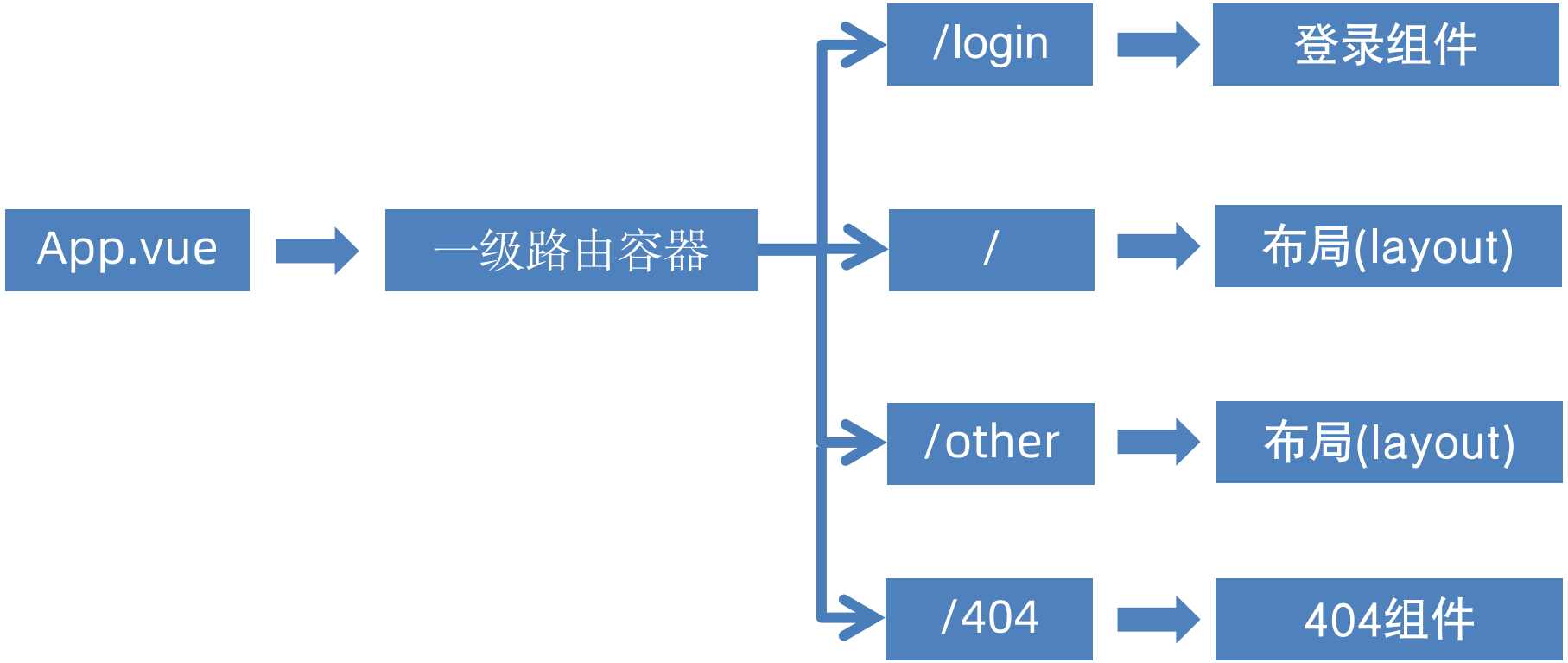
项目目录和文件介绍

```
├── src                         # 源代码目录
|   ├── api                     # 所有请求
|   ├── assets                  # 主题 字体等静态资源
|   ├── components              # 全局公用组件
|   ├── icons                   # 项目所有 svg icons
|   ├── layout                  # 全局 layout
|   ├── router                  # 路由
|   ├── store                   # 全局 store管理
|   ├── styles                  # 全局样式
|   ├── utils                   # 全局公用方法
|   ├── views                   # views 所有页面
|   ├── App.vue                 # 入口页面
|   ├── main.js                 # 入口文件 加载组件 初始化等
|   └── permission.js           # 权限管理
|   └── settings.js             # 配置文件
```

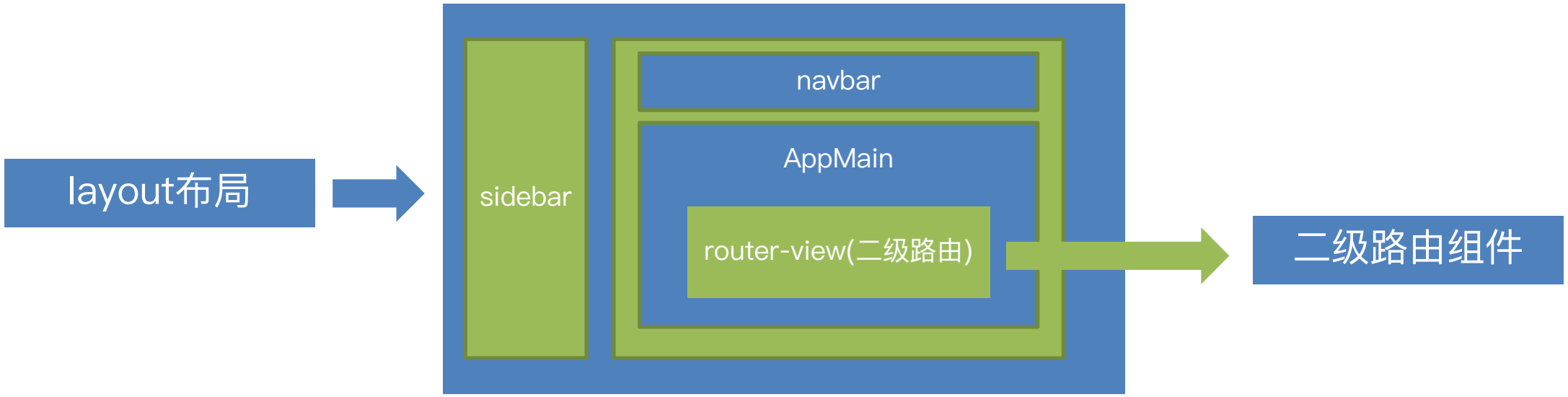
入口文件说明



根组件App.vue


App.vue
一级路由容器
/login
登录组件
/
布局(layout)
/other
布局(layout)
/404
404组件

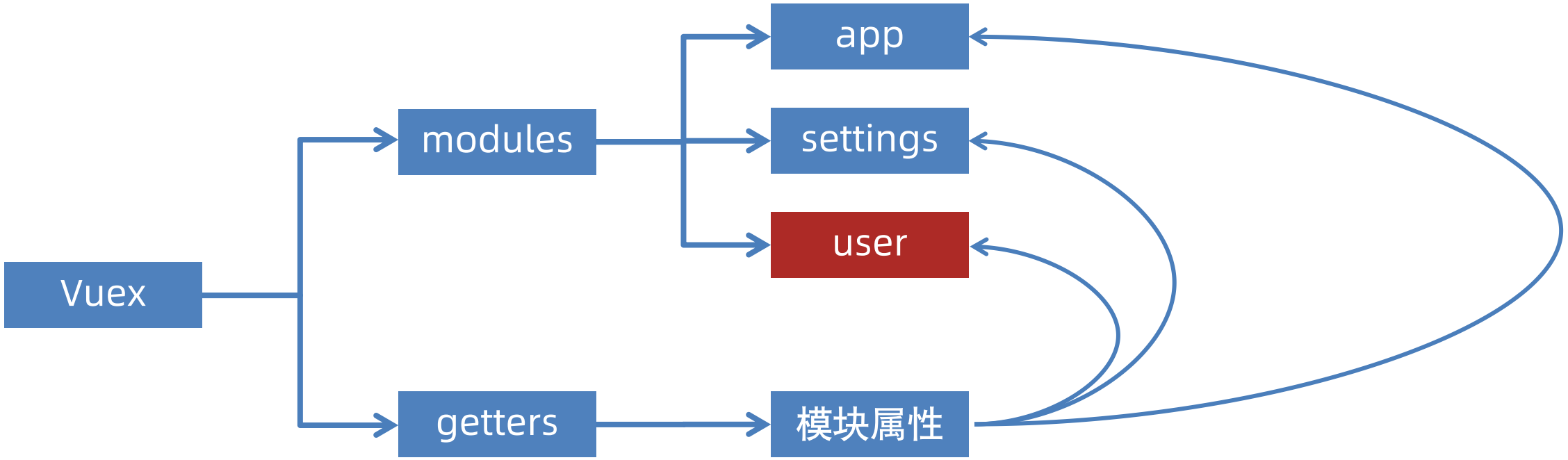
layout布局组件



基础设置settings.js和导航守卫permission.js

- settings.js导出网站基础配置，包括：网站标题、固定header、显示logo
- permission.js(权限)，主要负责路由导航守卫

Vuex结构



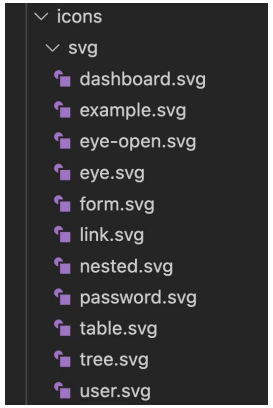
人力资源项目针对user模块重写，其他模块进行复用

模板中的Vuex的设计思想

- 页面交互状态（折叠侧边栏–固定头部）使用全局状态Vuex
- 根据功能拆分成不同的模块（modules）进行状态管理
- 通过getters建立对于模块中属性的快捷访问

使用模板中的icon图标

- src/icons/svg目录下的图标都可以使用



- 图标文件名直接设置为svg-icon组件的iconClass属性

```
<svg-icon icon-class="dashboard" />
<svg-icon icon-class="example" />
<svg-icon icon-class="eye-open" />
```

模板icons图标实现思路

模板Icons

全局注册svg-icon组件

```
// register globally
Vue.component('svg-icon', SvgIcon)
```

引入目录下所有svg

```
const req = require.context('./svg', false, /\.svg$/)
const requireAll = requireContext => requireContext.keys().map(requireContext)
requireAll(req)
```

loader插件打包svg

```
// set svg-sprite-loader
config.module
  .rule('svg')
  .exclude.add(resolve('src/icons'))
  .end()
config.module
  .rule('icons')
  .test(/\.svg$/)
  .include.add(resolve('src/icons'))
  .end()
  .use('svg-sprite-loader')
  .loader('svg-sprite-loader')
  .options({
    symbolId: 'icon-[name]'
  })
  .end()
```

svg-icon引用svg链接

```
<!-- 使用svg-icon组件  -->
<!-- 设置iconClass属性="文件名(没有后缀)" -->
<svg-icon icon-class="dashboard" />
<svg-icon icon-class="example" />
<svg-icon icon-class="password" />
<svg-icon icon-class="skill" />
```

知识扩展

- 了解：require.context(路径，是否扫描子目录， 正则匹配) 可以引入某个目录下的内容
- 了解：svg-sprite-loader打包了所有svg到一个svg标签上，将svg名称作为symbol标签的id属性
- 了解：svg-icon使用iconClass属性引用了symbol的id

导入样式和资源，使用git管理项目

- 将准备好的 图片，样式，svg导入到项目src中

- 删除原有git文件，重新初始化仓库，建立远程仓库，并同步推送

```
git init    # 初始化仓库
git add .   # 添加到暂存区
git commit -m “日志消息”  # 提交本地仓库
```

本地仓库

```
git remote add origin 远程仓库地址    # 本地仓库配置远程仓库地址
git push -u origin master    # 推送到远程仓库
```

推送远程

登录页的结构和表单

- 准备登录页面的样式和结构
- 实现登录表单的结构


```
<el-form autocomplete="off">
  <el-form-item>
    <el-input placeholder="请输入用户名" />
  </el-form-item>
  <el-form-item>
    <el-input show-password placeholder="请输入密码" />
  </el-form-item>
  <el-form-item>
    <el-checkbox>用户平台使用协议</el-checkbox>
  </el-form-item>
  <el-form-item>
    <el-button style="width: 350px" type="primary" block>登录</el-button>
  </el-form-item>
</el-form>
```

- 提交代码

登录表单校验-实现


```
<el-card shadow="never" class="login-card">
  <el-form ref="form" autocomplete="off" :rules="loginRules" :model="loginForm">
    <el-form-item prop="mobile">
      <el-input v-model="loginForm.mobile" placeholder="请输入用户名" />
    </el-form-item>
    <el-form-item prop="password">
      <el-input
        v-model="loginForm.password"
        show-password
        placeholder="请输入密码"
      />
    </el-form-item>
    <el-form-item prop="isAgree">
      <el-checkbox v-model="loginForm.isAgree">用户平台使用协议</el-checkbox>
    </el-form-item>
    <el-form-item>
      <el-button style="width: 350px" type="primary">登录</el-button>
    </el-form-item>
  </el-form>
```

登录表单的规则

- 手机号 必填，手机号规则

```
{ required: true, trigger: 'blur', message: '手机号不能为空' },
{ trigger: 'blur', pattern: /^1[3-9]\d{9}$/, messageL: '手机号格式不正确' }
```

- 密码 必填，6-16位长度

```
{ required: true, trigger: 'blur', message: '密码不能为空' },
{ min: 6, max: 16, message: '密码的长度在6-16位之间', trigger: 'blur' }
```

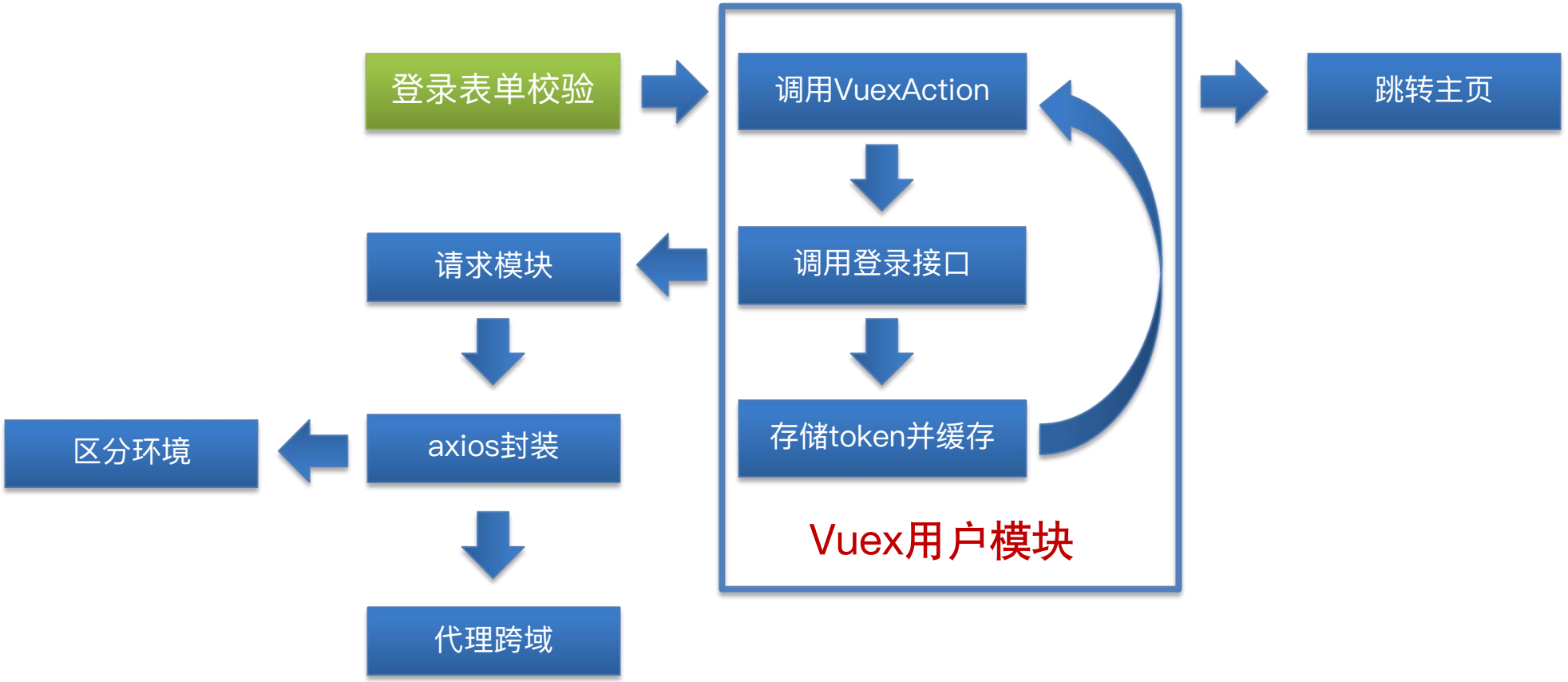
- 用户协议 必须勾选

```
isAgree: [{
  validator: (rule, value, callback) =>
    value ? callback() : callback(new Error('您还未勾选用户协议'))
}
```

- 表单整体校验


```
login() {
  this.$refs.form.validate(async(isOK) => {
    if (isOK) {
      alert('校验通过')
    }
  })
}
```

分析登录流程


登录表单校验
调用VuexAction
跳转主页
请求模块
调用登录接口
区分环境
axios封装
存储token并缓存
代理跨域
Vuex用户模块

Vuex的用户模块实现

- 在src/store/modules/user.js中管理token

```
const state = {
  token: null
}
const mutations = {}
const actions = {}

export default {
  namespaced: true,
  state,
  mutations,
  actions
}
```

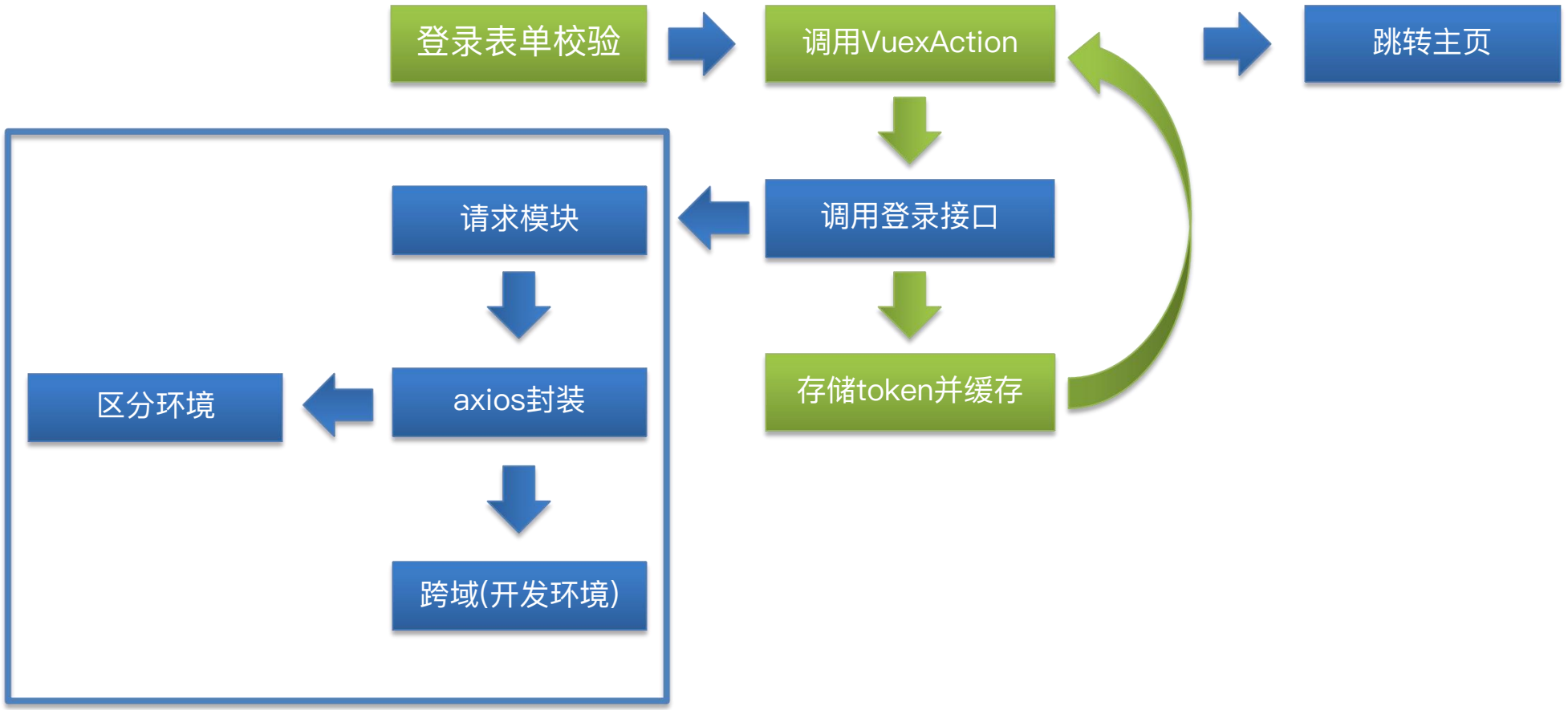
```
import { getToken, setToken, removeToken } from '@/utils/auth'

const state = {
  token: getToken()
}
const mutations = {
  setToken(state, token) {
    state.token = token // 只是设置了vuex中的数据
    // 需要将vuex中的数据同步到缓存
    setToken(token)
  },
  removeToken(state) {
    state.token = null // 设置vuex中的token为null
    removeToken() // 同步删除缓存中的token
  }
}
```

- 实现token的Vuex数据持久化
- 实现登录的action方法

```
async login(context, data) {
  // todo: 调用登录接口
  context.commit('setToken', '12345')
}
```

```
if (isOK) {
  this.$store.dispatch('user/login', this.loginForm)
}
```

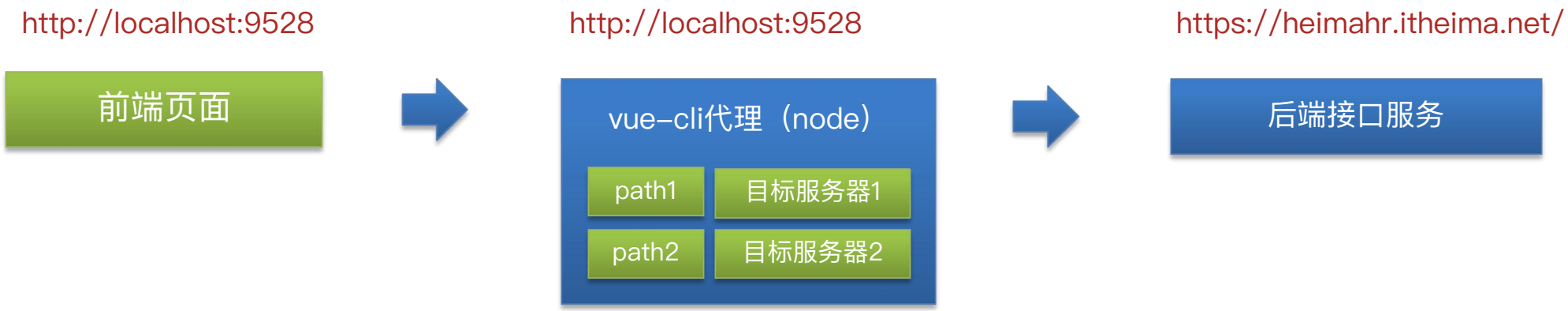
请求分析


登录表单校验
调用VuexAction
跳转主页
调用登录接口
请求模块
存储token并缓存
axios封装
区分环境
跨域(开发环境)

vue-cli代理解决跨域

- 直接使用前端向后端发请求（后端并没有开启cors）

- 代理解决方案



vue-cli配置跨域

- vue.config.js（改完要重启）

```
proxy: {
  '/api': {
    target: 'http://heimahr.itheima.net' // 要代理的目标地址
  }
}
```

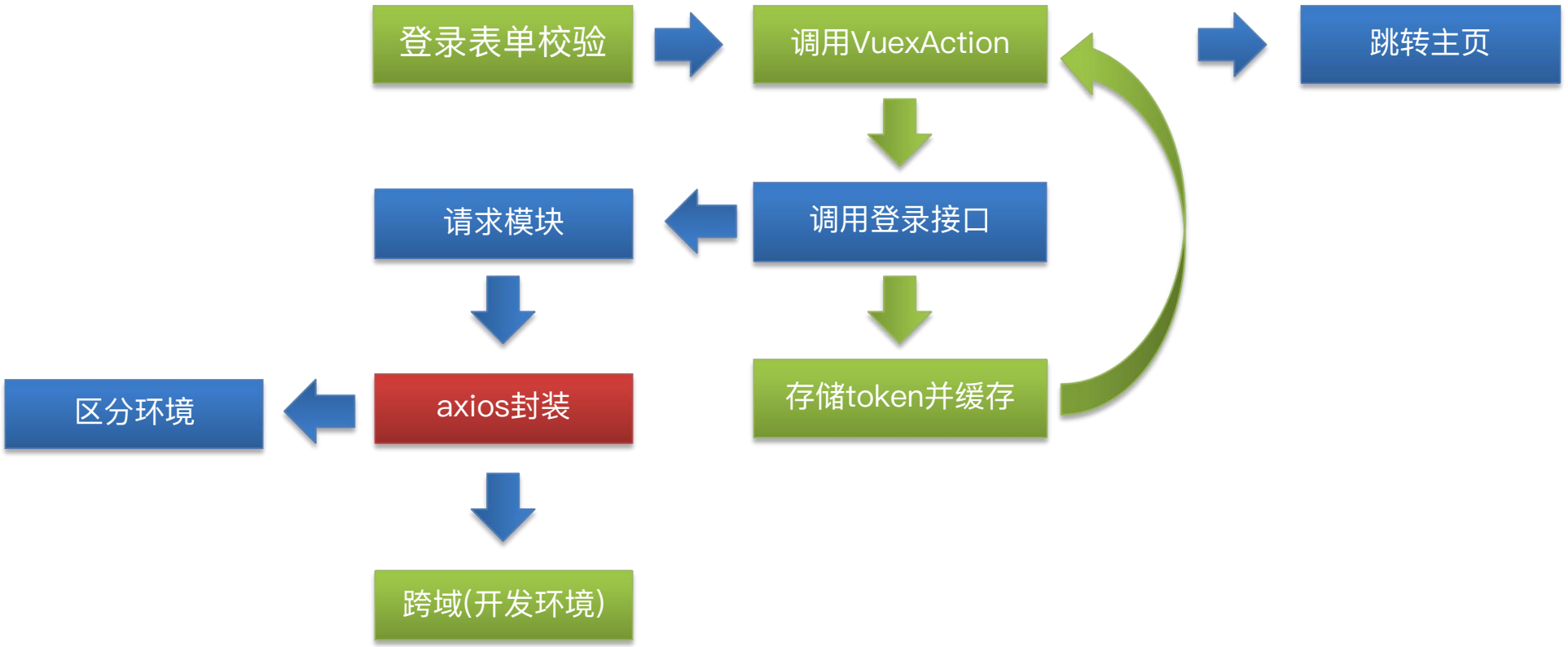
- 注意删除原有的before配置选项

```
    }
    // before: require('./mock/mock-server.js')
  },
```

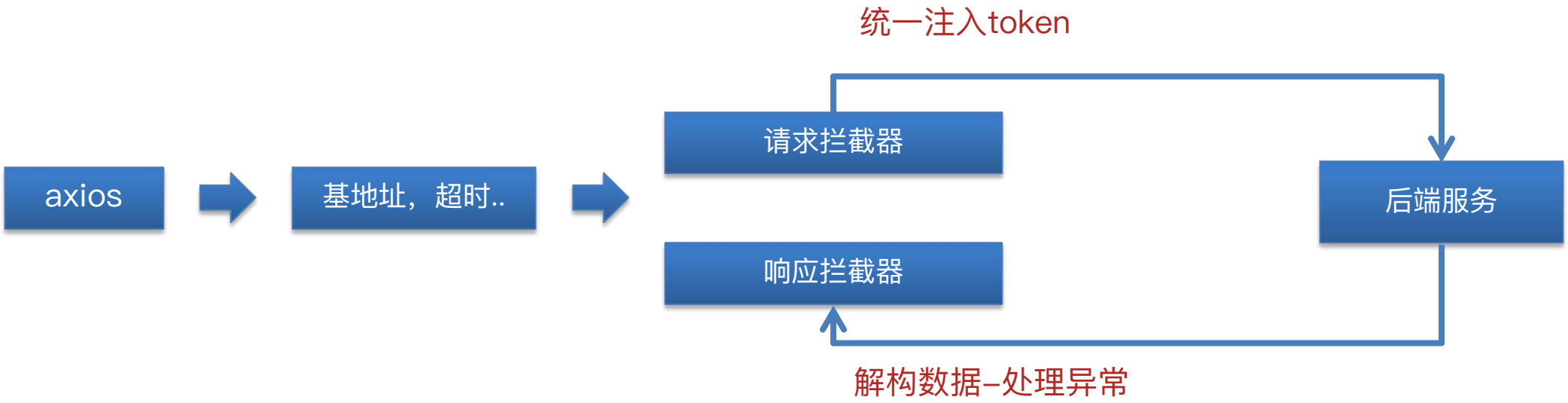
- 发请求验证

```
axios({
  method: 'post',
  // url: 'https://heimahr.itheima.net/api/sys/login',
  url: '/api/sys/login',
  data: {
    mobile: '13912345678',
    password: '123456'
  }
})
```

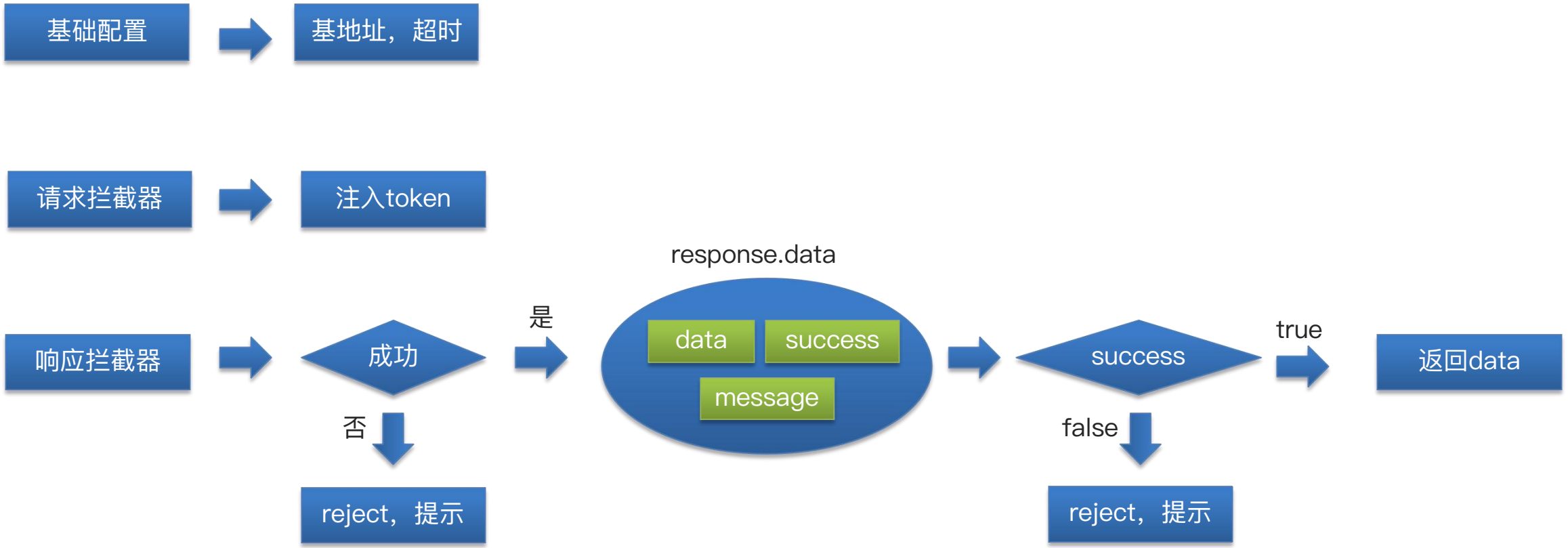
axios封装


登录表单校验
调用VuexAction
跳转主页
调用登录接口
请求模块
存储token并缓存
axios封装
区分环境
跨域(开发环境)

axios封装的需求



axios功能


基础配置
基地址，超时
请求拦截器
注入token
响应拦截器
成功
否
reject，提示
是
response.data
data
success
message
success
true
返回data
false
reject，提示

axios封装总结

基础配置

```
const service = axios.create({
//     设置基础地址
  baseURL: '/api',
  timeout: 10000
})
```

响应拦截器

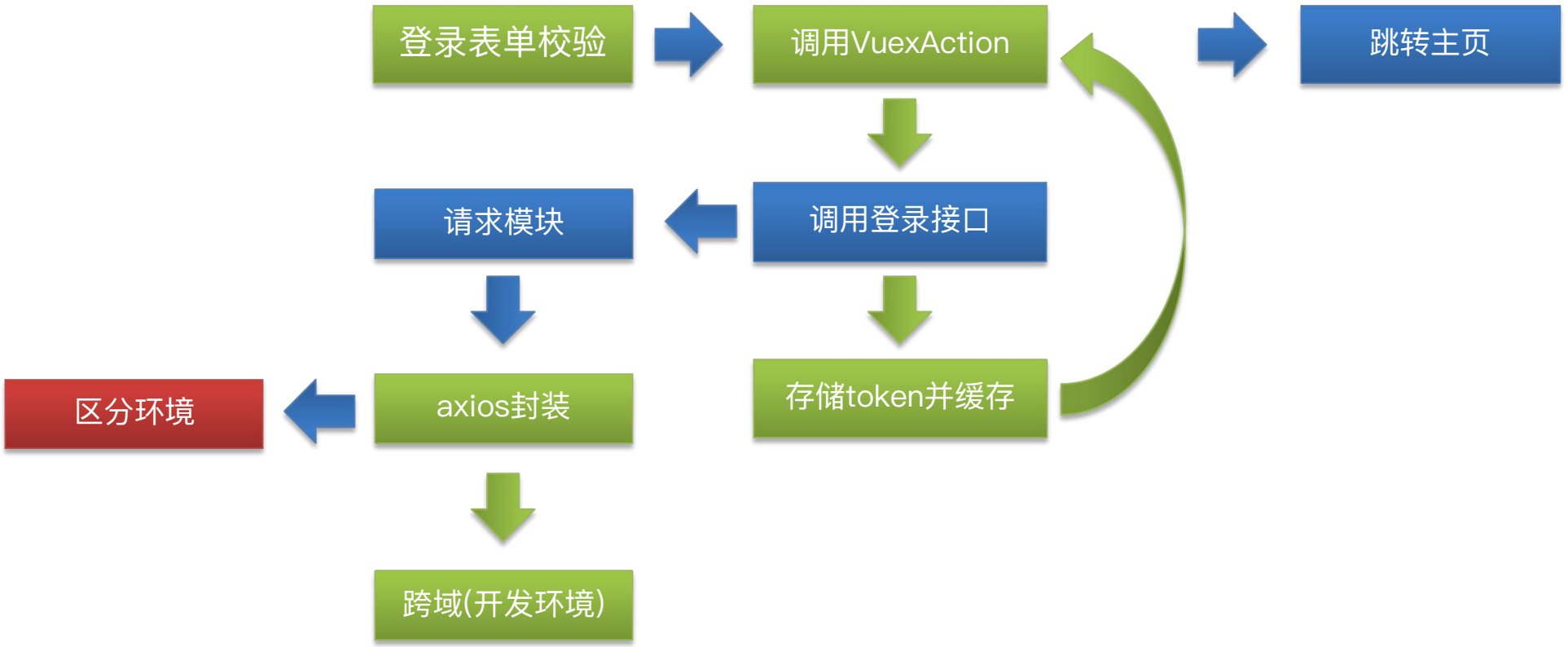
```
service.interceptors.response.use(response => {
  // 成功执行
  // axios默认加了一层data的包裹
  const { success, message, data } = response.data
  if (success) {
    // 此时认为业务执行成功了
    return data // 返回用户所需要的数据
  } else {
    // 当业务失败的时候
    Message.error(message) // 提示消息
    return Promise.reject(new Error(message))
  }
}, async error => {
  Message.error(error.message) // 提示错误
  // reject
  return Promise.reject(error)
})
```

请求拦截器

```
// 请求拦截器
service.interceptors.request.use(async config => {
  // 请求接口  config是请求配置
  // 取token
  if (store.getters.token) {
    // 只要有token 就要检查token时效性
    // 如果存在token
    config.headers.Authorization = `Bearer ${store.getters.token}`
    // return config
  }
  // 这里一定要注意
  // 一定要return config
  return config
}, error => {
  return Promise.reject(error)
})
```

人力资源接口文档

环境区分


登录表单校验
调用VuexAction
跳转主页
调用登录接口
请求模块
存储token并缓存
axios封装
区分环境
跨域(开发环境)

区分环境

开发环境 → yarn dev → 开启本地服务

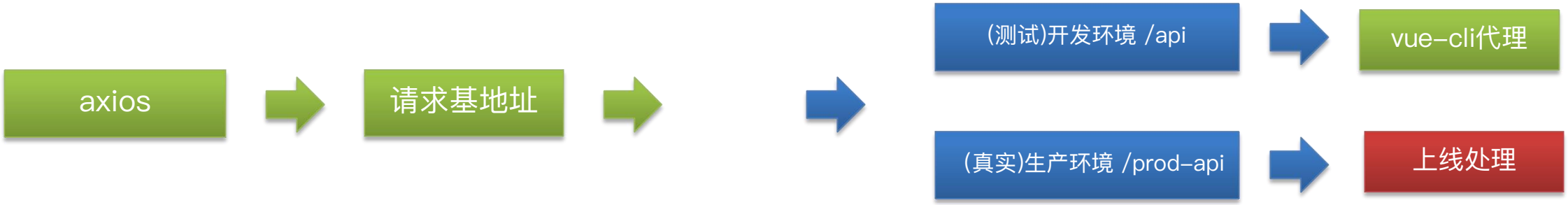
生产环境 → yarn build:prod → 打包html+css+js → 服务器部署

.env.development中设置开发环境变量 默认 NODE_ENV 值为 development

.env.production中设置生产环境变量 默认 NODE_ENV 值为 production

使用process.env.属性的方式获取

axios的请求处理基地址

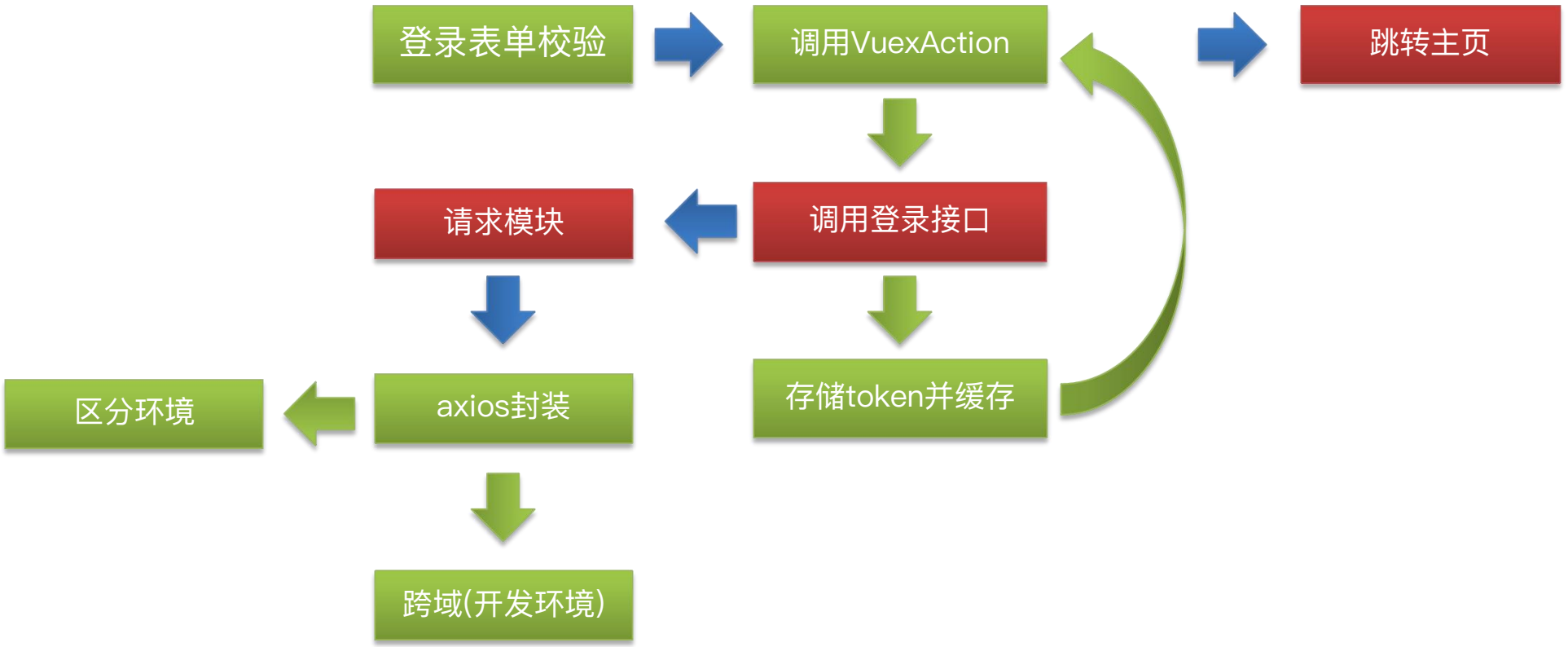


Vue代码中NODE_ENV之外，所有的变量必须以VUE_APP_开头

```
# base api
VUE_APP_BASE_API = '/api'
```

```
const service = axios.create({
  baseURL: process.env.VUE_APP_BASE_API, // 基础地址
  timeout: 10000
}) // 创建一个新的axios实例
```

登录联调


登录表单校验
调用VuexAction
跳转主页
调用登录接口
请求模块
存储token并缓存
axios封装
区分环境
跨域(开发环境)

登录联通

1.封装API请求

```
import request from '@/utils/request'

export function login(data) {
  // 返回了promise对象
  return request({
    url: '/sys/login',
    method: 'post',
    data // body参数体位于data
  })
}
```

2.Vuex调用

```
async login(context, data) {
  const result = await login(data)
  // result就是token
  context.commit('setToken', result)
},
```

3.跳转主页

```
async login() {
  this.$refs.form.validate(async isOK => {
    if (isOK) {
      await this.$store.dispatch('user/login', this.loginForm)
      this.$router.push('/')
    }
  })
},
```

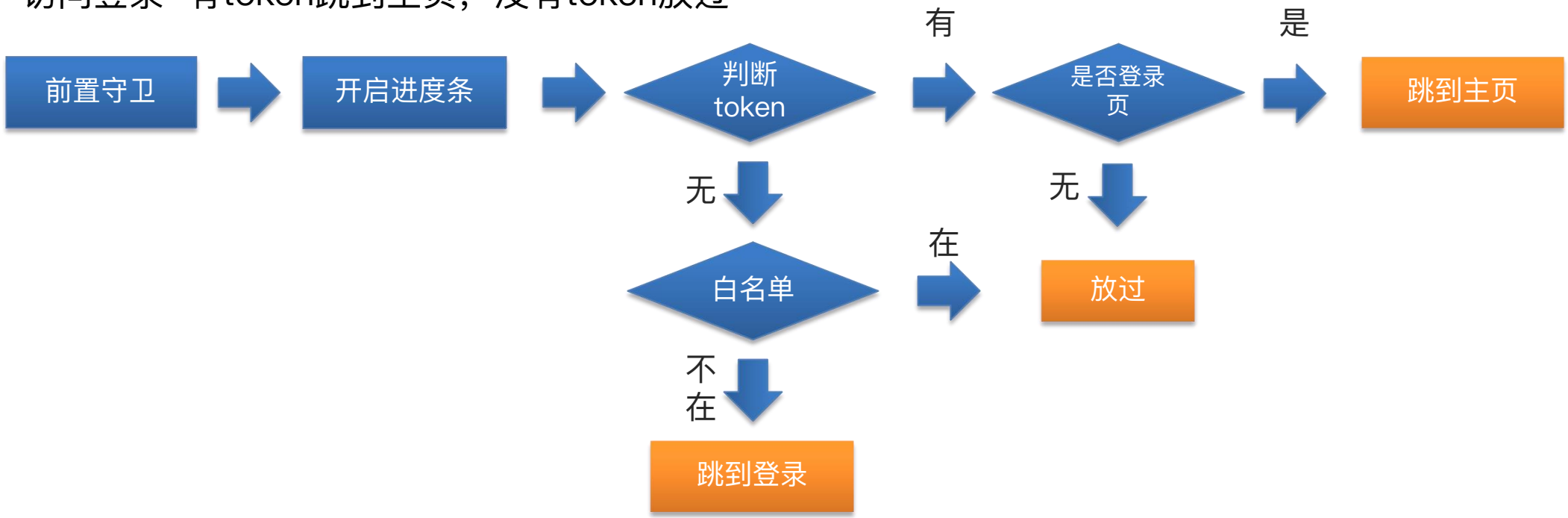
4.区分环境数据

```
loginForm: {
  mobile: process.env.NODE_ENV === 'development' ? '13800000002' : '',
  password: process.env.NODE_ENV === 'development' ? '123456' : '',
  isAgree: process.env.NODE_ENV === 'development'
},
```

主页权限验证(permission.js)

访问主页–有token放过，没有token跳到登录页

访问登录–有token跳到主页，没有token放过



权限实现

```
// 前置守卫
const whileList = ['/login', '/404']
router.beforeEach(async(to, from, next) => {
  NProgress.start() // 开启进度条
  // next是一个必须执行的钩子 不执行就卡主了
  if (store.getters.token) {
    if (to.path === '/login') {
      next('/') // 跳到主页1
      NProgress.done() // 是为了解决手动输入地址时 进度条不关闭的问题
    } else {
      next() // 放行 1. 有token的情况下判断 是不是登录页
    }
  } else {
    if (whileList.indexOf(to.path) > -1) {
      // 表示在白名单里面
      next()
    } else {
      next('/login')
      NProgress.done() // 是为了解决手动输入地址时 进度条不关闭的问题
    }
  }
})
```

2. 没token的情况下 判断在不在白名单

```
// 后置守卫
router.afterEach(() => {
  NProgress.done()
})
```

传智教育旗下高端IT教育品牌