优医问诊day02

登录模块、个人中心

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

路由规则配置

路由规则

登录　　极速问诊

布局容器　　...

首页　　健康百科

消息通知　　个人中心

准备登录路由和组件

组件自动注册

为何需要组件自动注册？

使用 Vant 组件较麻烦，需要先导入组件才可使用

```
<template>
  <div class="login-page">
    <van-button>按钮</van-button>
    <van-checkbox>复选框</van-checkbox>
  </div>
</template>
```

组件自动注册：无需在 script setup 中导入，直接使用 （不是全局注册）

如何配置组件自动注册？

1. 安装：pnpm add unplugin-vue-components -D
2. 配置：vite.config.ts 参考文档

```ts
import Components from 'unplugin-vue-components/vite'
import { VantResolver } from 'unplugin-vue-components/resolvers'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [
    vue(),
    Components({
      dts: false,
      resolvers: [VantResolver({ importStyle: false })]
    })
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url))
    }
  }
})
```

配置：

1. dts: false 不自动生成组件类型文件
   导入包后就能识别
2. importStyle: false 不自动导入样式
   已经在 main.ts 导入

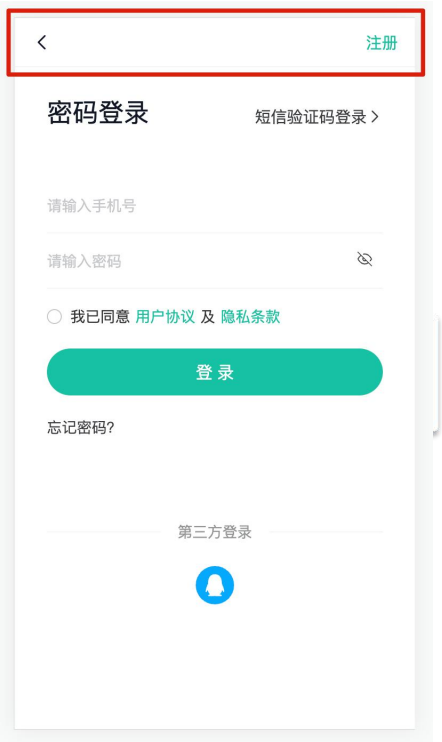
默认 components 文件下也会自动注册

通用组件cp-nav-bar封装
结构与样式

通用组件 CpNavBar 封装

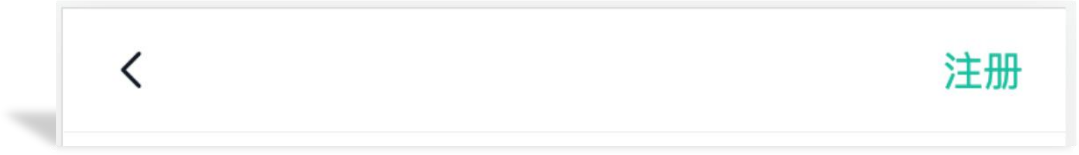
- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-结构与样式



- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-结构与样式



结构：

1. 固定定位
2. 左侧箭头
3. 标题
4. 右侧文字

参数	说明
title	标题
left-text	左侧文案
right-text	右侧文案
left-arrow	是否显示左侧箭头
border	是否显示下边框
fixed	是否固定在顶部

样式：

1. 左侧箭头 --cp-text1 颜色，18px 字体
2. 标题 15px 字体

```
<style lang="scss" scoped>
:deep() {
  0 references
  .van-nav-bar {…
  }
}
</style>
```

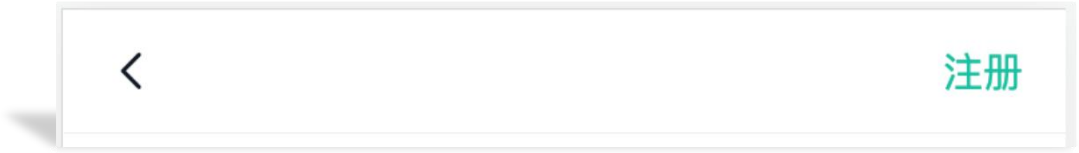
04

通用组件cp-nav-bar封装
标题和右侧文字&右侧点击事件

通用组件 CpNavBar 封装-功能实现

- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-功能实现



props 绑定属性

- 标题：title 可选
- 右侧文字：right-text 可选

emit 自定义事件

- 点击右侧文字：click-right (没有文字时候无效)

通用组件cp-nav-bar封装回退功能

通用组件 CpNavBar 封装-功能实现

- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-功能实现

- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-功能实现

回退功能：

1. 有访问历史记录就正常回退 router.back()
2. 没有访问历史记录跳转首页 router.push('/')


```
history.state
▶ {back: null, current: '/home', forward: null, position: 1, replaced: true, …}
history.state
▶ {back: '/home', current: '/login', forward: null, replaced: false, position: 2, …}
```

通用组件cp-nav-bar封装
组件类型

通用组件 CpNavBar 封装-组件类型

- **结构与样式**：基于 Vant 组件二次封装，实现组件结构，覆盖组样式
- **功能实现**：支持标题和右侧文字，提供点击右侧文字按钮事件，实现返回功能
- **组件类型**：提供组件类型文件，让组件有类型提示

通用组件 CpNavBar 封装-组件类型

局部组件：

- 显性的导入组件，会自动推导出组件类型

全局组件、自动注册的组件：

- 没有显性导入组件，需要添加全局组件类型

types/components.d.ts

```ts
import CpNavBar from '@/components/CpNavBar.vue'

declare module 'vue' {
  interface GlobalComponents {
    CpNavBar: typeof CpNavBar
  }
}
```

全局组件类型：

1. declare module `vue` 扩展模块类型
2. interface GlobalComponents 扩展全局组件
3. typeof 根据 JS变量（组件对象） 得到类型

07 登录页面–页面布局

页面布局

重置样式

准备页面

定制表单

mount API 变化：

- vue2.x 中 App组件 替换 #app 容器
- vue3.x 中 App组件 插入 #app 容器

08 登录页面–表单校验

登录页面-表单校验

校验手机号

校验密码

表单整体校验

校验勾选协议

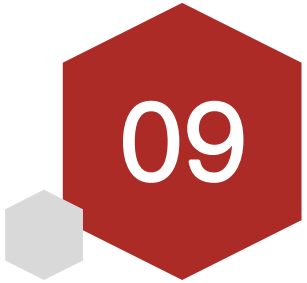
```ts
import type { FieldRule } from 'vant'

// 表单校验
const mobileRules: FieldRule[] = [
  { required: true, message: '请输入手机号' },
  { pattern: /^1[3-9]\d{9}$/, message: '手机号格式不正确' }
]

const passwordRules: FieldRule[] = [
  { required: true, message: '请输入密码' },
  { pattern: /^\w{8,24}$/, message: '密码需8-24个字符' }
]

export { mobileRules, passwordRules }
```

utils/rules.ts

```vue
<van-button native-type="submit" block round type="primary">
```

```vue
<van-form autocomplete="off" @submit="onSubmit">
```

```ts
const agree = ref(false)
const onSubmit = () => {
  if (!agree.value) return showToast('请勾选用户协议')
  // TODO 进行登录
}
```



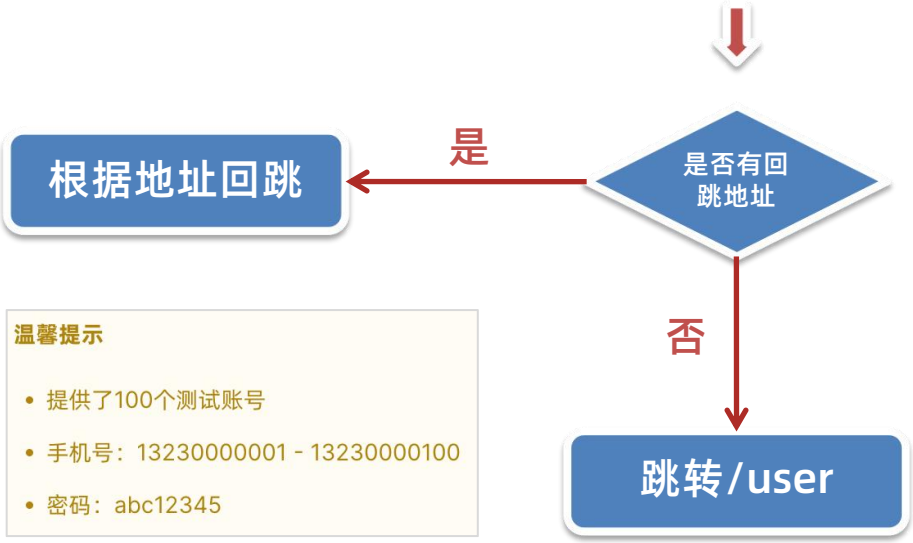
登录页面–密码登录

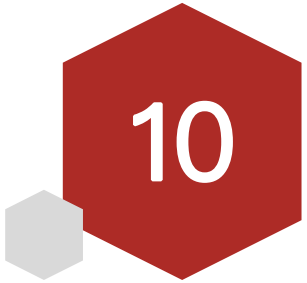
登录页面-密码登录

登录API函数 → 进行登录 → 存储用户信息 → 成功提示

```ts
export const loginByPassword = (mobile: string, password: string) =>
  request<User>('login/password', 'POST', { mobile, password })
```

```ts
const router = useRouter()
const route = useRoute()
const store = useUserStore()
const onSubmit = async () => {
  if (!agree.value) return showToast('请勾选协议')
  // 进行登录
  const res = await loginByPassword(mobile.value, password.value)
  // 成功：存储用户信息+成功提示+跳转地址
  store.setUser(res.data)
  showSuccessToast('登录成功')
  router.replace((route.query.returnUrl as string) || '/user')
}
```





登录页面–短信登录–切换效果

短信登录

短信登录

登录页面-短信登录-切换界面

标题按钮切换

表单项切换

验证码校验

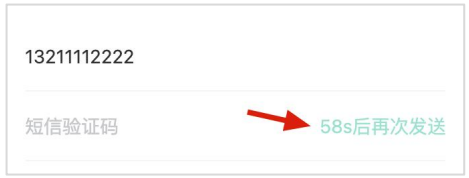
登录页面–短信登录–发送验证码

短信登录

短信登录-发送短信验证码

短信登录-发送短信验证码

发送前校验 → 验证码API函数 → 发送&提示 → 倒计时&展示




```ts
export type CodeType =
  | 'login'
  | 'register'
  | 'changeMobile'
  | 'forgetPassword'
  | 'bindMobile'
```

```ts
// 发送验证码
export const sendMobileCode = (mobile: string, type: CodeType) =>
  request('/code', 'GET', { mobile, type })
```

```ts
const form = ref<FormInstance>()
const time = ref(0)
const onSend = async () => {
  if (time.value > 0) return
  await form.value?.validate('mobile')
```

```ts
await sendMobileCode(mobile.value, 'login')
showToast('发送成功')
time.value = 60
```

```ts
  // 倒计时
  if (timer) clearInterval(timer)
  timer = setInterval(() => {
    time.value--
    if (time.value <= 0) clearInterval(timer)
  }, 1000)
}

onUnmounted(() => {
  clearInterval(timer)
})
```

```vue
<span class="btn-send" :class="{ active: time > 0 }" @click="onSend">
  {{ time > 0 ? `${time}s后再次发送` : '发送验证码' }}
</span>
```

发送短信验证码的基础流程，
以后遇到相近业务就可以复用

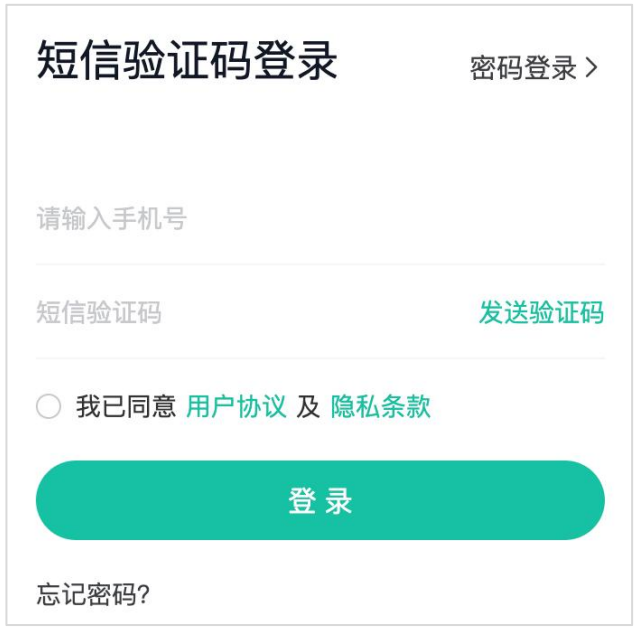
登录页面–短信登录–进行登录

短信登录

短信登录-进行登录

提示：开发中发送短信验证码是后台模拟的，并没有发送到手机上。
我们可以在响应报文中找到验证码进行登录。

短信登录-进行登录



短信登录API

合并短信登录

```
// 短信登录
export const loginByMobile = (mobile: string, code: string) =>
  request<User>('/login', 'POST', { mobile, code })
```

```
const res = isPass.value
  ? await loginByPassword(mobile.value, password.value)
  : await loginByMobile(mobile.value, code.value)
```

图标组件–打包svg地图

图标组件-什么是svg地图

图片格式png的：合成精灵图，根据定位使用

图片格式svg的：它是 xml 格式，打包成svg地图，通过ID来使用即可，一次加载，动态使用
图片无需 import

```
▶<symbol fill="none" viewBox="0 0 62 62" id="icon-home-rp">…</symbol>
▶<symbol fill="none" viewBox="0 0 34 34" id="icon-home-search">…</symbol>
▶<symbol fill="none" viewBox="0 0 32 32" id="icon-login-eye-off">…</symbol>
▶<symbol fill="none" viewBox="0 0 32 32" id="icon-login-eye-on">…</symbol>
▶<symbol fill="none" viewBox="0 0 40 40" id="icon-user-add">…</symbol>
▶<symbol fill="none" viewBox="0 0 40 40" id="icon-user-arrow">…</symbol>
```

```
▼<svg aria-hidden="true" class="cp-icon" data-v-e007b029 data-v-58e7910a>
   <use href="#icon-login-eye-off" data-v-e007b029></use>
 </svg>
</div>
```

图标组件-打包svg地图

安装：pnpm install vite-plugin-svg-icons -D

配置：vite.config.ts

```
// 配置svg插件
import { createSvgIconsPlugin } from 'vite-plugin-svg-icons'
import path from 'path'
```

```
createSvgIconsPlugin({
  iconDirs: [path.resolve(process.cwd(), 'src/icons')]
})
```

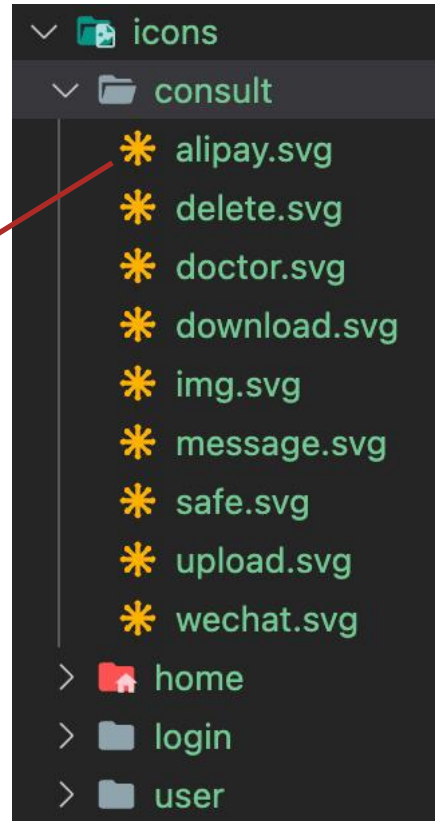
导入：main.ts

```
import 'virtual:svg-icons-register'
```

使用：

```
<svg aria-hidden="true">
  <use href="#icon-consult-alipay" />
</svg>
```

文件夹　　图标名称



14 图标组件–封装svg组件

图标组件-封装svg组件

```vue
<svg aria-hidden="true">
  <use href="#icon-consult-alipay" />
</svg>
```

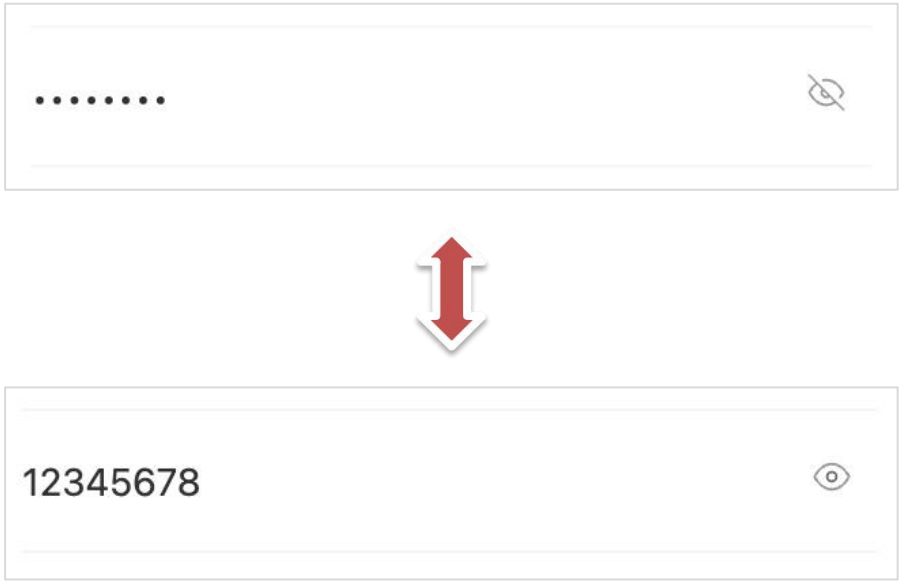
```vue
<cp-icon name="consult-alipay" />
```

```ts
import CpNavBar from '@/components/CpNavBar.vue'
import CpIcon from '@/components/CpIcon.vue'

declare module 'vue' {
  interface GlobalComponents {
    // 添加组件类型
    CpNavBar: typeof CpNavBar
    CpIcon: typeof CpIcon
  }
}
```

```vue
<script setup lang="ts">
defineProps<{
  name: string
}>()
</script>

<template>
  <svg aria-hidden="true" class="cp-icon">
    <use :href="`#icon-${name}`" />
  </svg>
</template>

<style lang="scss" scoped>
1 reference
.cp-icon {
  width: 1em;
  height: 1em;
} 和字体大小一致
</style>
```

通过 name 设置

实现密码可见与不可见




```
const isShow = ref(false)
```

```
<van-field
  v-if="isPass"
  v-model="password"
  :rules="passwordRules"
  placeholder="请输入密码"
  :type="`${isShow ? 'text' : 'password'}`"
>
  <template #button>
    <cp-icon
      @click="isShow = !isShow"
      :name="`login-eye-${isShow ? 'on' : 'off'}`"
      style="margin-right: 10px"
    ></cp-icon>
  </template>
</van-field>
```

传智教育旗下高端IT教育品牌