

微信小程序开发



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

01

分包加载

- 为什么分包?
- 使用分包
- 分包预加载

小程序分包加载（按需加载）

- 为什么进行分包加载？
 - 小程序限制单个代码包体积不超过2M
 - 分包可以优化小程序页面的加载速度
- 启用/使用分包：subPackages
 - root 分包所在的目录
 - pages 分包中包含的页面
 - tabBar 的页面不允许分包
 - 所有包不超过20M
- 分包预加载：preloadRule
 - 页面路径做为 key（属性）
 - network 预加载的网络环境
 - packages 需要预加载的包

```
"packOptions": {  
  "ignore": [  
    { "value": "static/uploads", "type": "folder" }  
  ],  
  "include": []  
},
```

```
"subPackages": [  
  {  
    "root": "subpkg_test",  
    "pages": ["pages/gallery/index"]  
  },  
  {  
    "root": "subpkg_user",  
    "pages": ["pages/profile/index"]  
  }  
],
```

```
"preloadRule": {  
  "pages/framework/index": {  
    "network": "all",  
    "packages": ["subpkg_user"]  
  }  
},
```

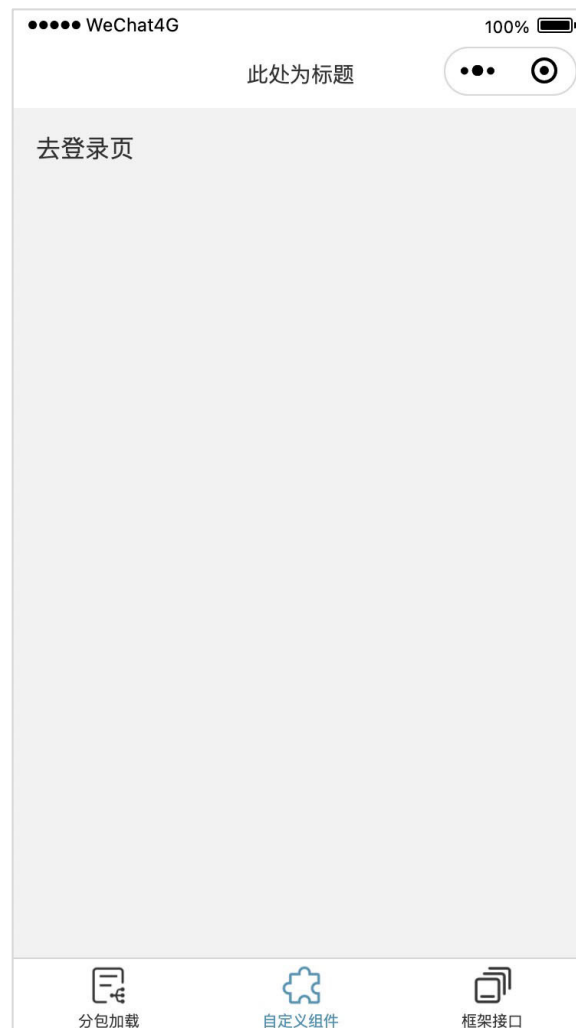
02

自定义组件

- 为什么分包?
- 使用分包
- 分包预加载

小程序自定义组件 - 基本语法

- 创建自定义组件
 - .json 文件中 `component: true`
 - .js 中调用 `Component` 函数
- 使用自定义组件
 - 全局配置或页面配置 `usingComponents`
 - 页面中以组件形式引入



本节知识总结:

- 创建自定义组件:
 - .json 文件中配置 `component: true`

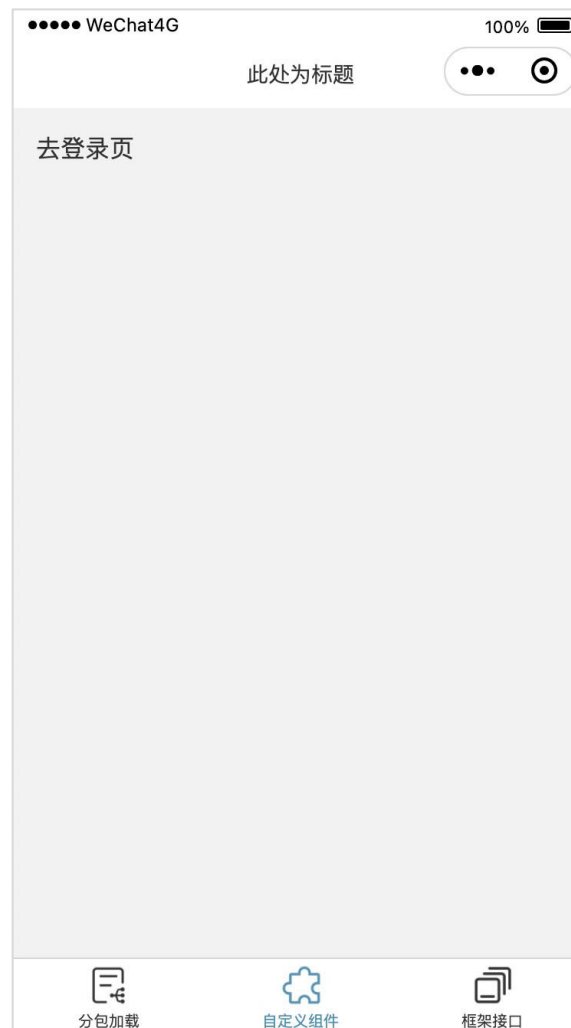
```
{  
  "component": true,  
  "usingComponents": {}  
}
```

- .js 文件中调用 `Component` 函数
- 使用自定义组件:
 - 全局或页面配置 `usingComponents`

```
"usingComponents": {  
  "navigation-bar": "./navigation-bar/index"  
},
```

小程序自定义组件 - 组件样式

- 样式隔离
 - .js 文件中传入 `options: { addGlobalClass: true }`
 - 尽量不要使用标签、ID、属性选择器
- 外部样式类
 - .js 文件中传入 `externalClasses: [ xxx, yyy ]`
 - xxx、yyy 可以理解成 “变量”



本节知识总结:

- 样式隔离:

- 尽量不要使用标签、ID、属性选择器

- ```
options: {
 addGlobalClass: true
},
```

- 外部样式类:

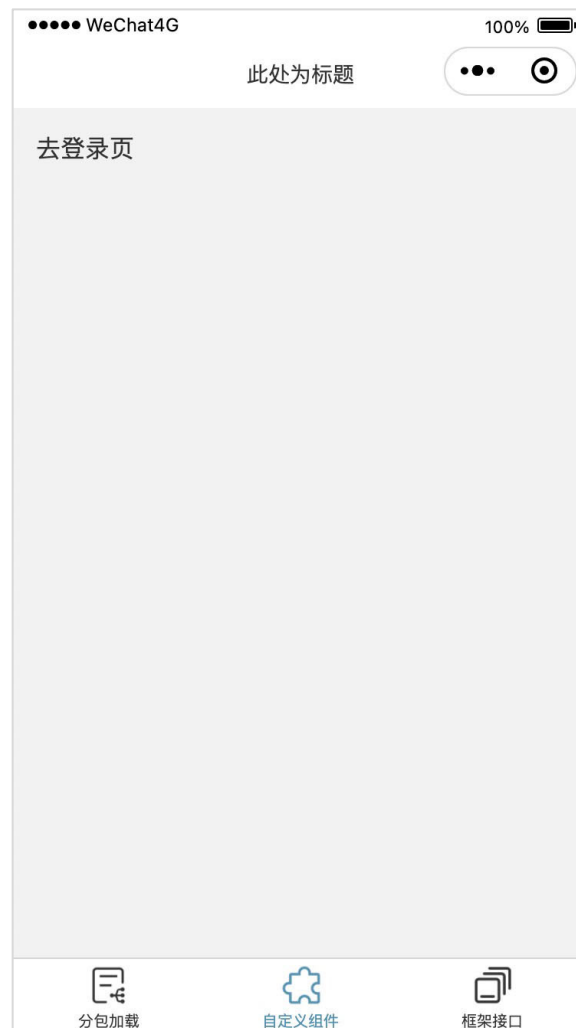
- ```
// 外部样式类  
externalClasses: ['custom-class', 'title-class'],
```

- ```
<view class="navigation-bar custom-class">
 <view class="navigation-bar-title title-class">
 自定义标题
 </view>
</view>
```

```
<navigation-bar
 title-class="my-navigation-bar-title"
 custom-class="my-navigation-bar"
></navigation-bar>
```

## 小程序自定义组件 - <slot /> (插槽)

- 创建插槽：
  - 默认情况只能 1 个 <slot />
  - .js 文件中传入 `options: { multipleSlots: true }`
  - `<slot name= "title" />` 为插槽命名
- 使用插槽
  - 单个插槽：在组件中间填充内容
  - 多个插槽：使用 slot 属性指定插槽位置



## 本节知识总结:

- 开启多插槽:

- ```
options: {  
  addGlobalClass: true,  
  multipleSlots: true  
},
```

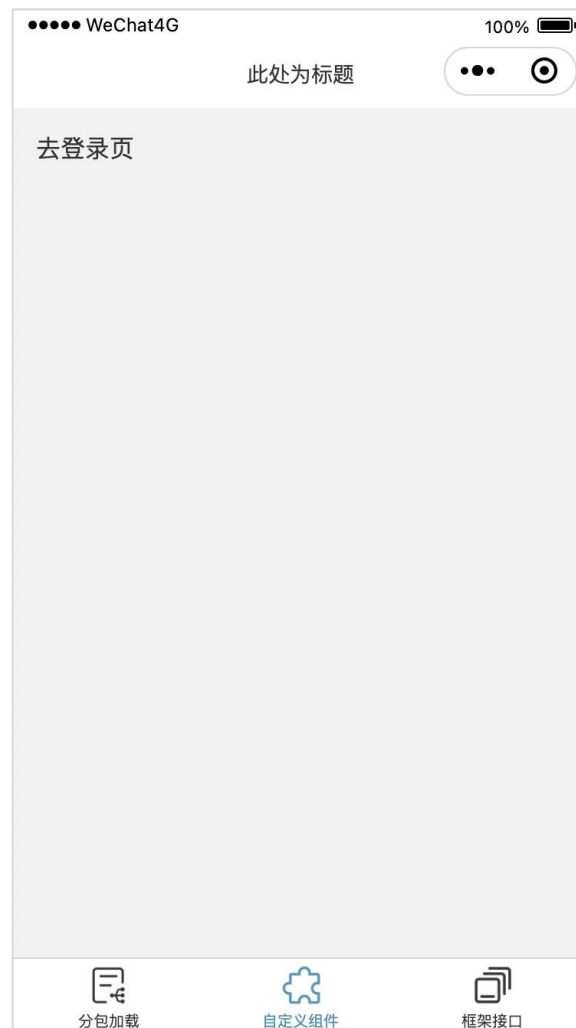
- ```
<!-- 定义插槽 -->
<slot name="title"></slot>
```

- 使用插槽

- ```
<!-- 使用插槽 -->  
<view slot="title">自定义标题</view>
```

小程序自定义组件 - lifetimes（生命周期）

- **created**：组件创建时触发
 - 类似 Vue 中的 created
 - 不能调用 this.setData
- **attached**：组件初始完毕时触发
 - 类似于 Vue 中的 mounted
 - 最常使用
- 应用示例：状态栏适配
 - wx.getSystemInfoSync()

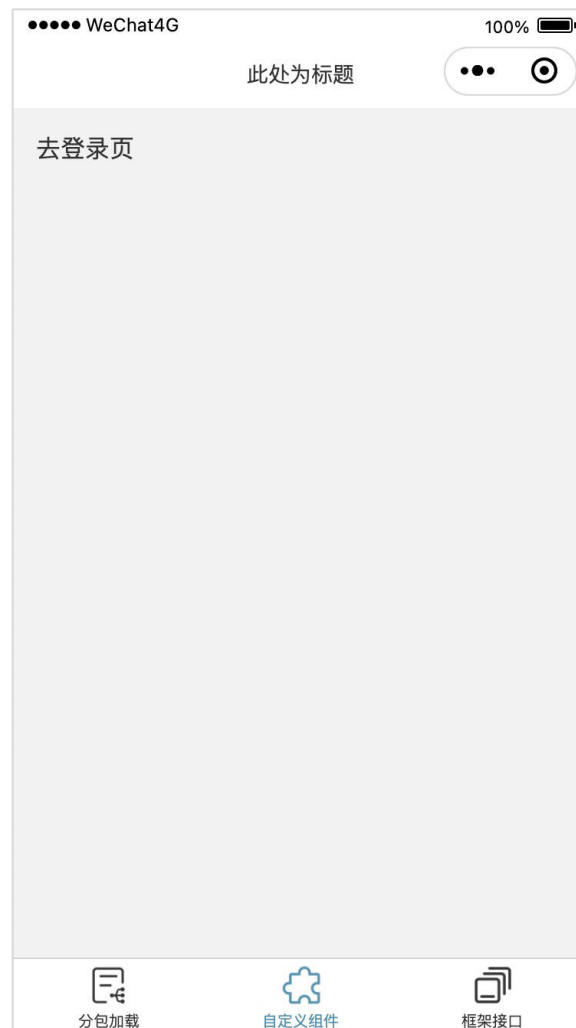


小程序自定义组件 - 组件通信

- 自定义属性: `properties`

```
properties: {  
  back: Boolean,  
  delta: {  
    type: Number,  
    value: 1  
  }  
},
```

- 应用示例: 路由返回
 - `wx.navigateBack()`



小程序自定义组件 - 组件通信

- 应用示例：倒计时组件
 - 外部样式类
 - `<slot />` 插槽
 - 自定义属性
 - 生命周期
- 自定义事件：bind:事件类型（自定义）
 - `this.triggerEvent('事件类型', 参数)`

```
<!-- 监听自定义事件 -->  
<count-down bind:change="countDownChange"></count-down>
```

```
// 传递数据到组件外部  
this.triggerEvent('change', this.data.time)
```



小程序自定义组件 - 组件通信

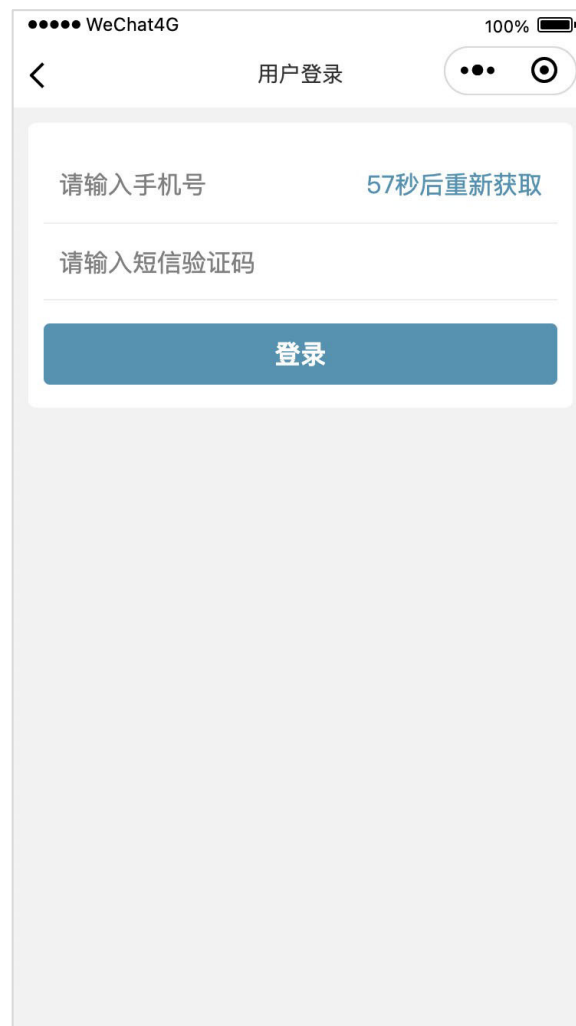
- 应用示例：倒计时组件
 - 外部样式类
 - `<slot />` 插槽
 - 自定义属性
 - 生命周期
- 自定义事件：bind:事件类型（自定义）
 - `this.triggerEvent('事件类型', 参数)`

```
<!-- 监听自定义事件 -->
```

```
<count-down bind:change="countDownChange"></count-down>
```

```
// 传递数据到组件外部
```

```
this.triggerEvent('change', this.data.time)
```



小程序自定义组件 - Vant 组件库

- 安装 Vant 组件库
 - npm install
 - npm 构建
 - 移除 style: "v2"
- Vant 组件演示
 - van-button
 - van-field
 - van-cell-group



03

框架接口

- 昵称和头像
- 应用实例
- 页面栈（页面实例）

小练习：昵称和头像

- 用户昵称：
 - input 组件的 type 属性设置为 **nickname**
 - 监听 **blur** 事件获取用户昵称
 - 存储用户昵称
- 用户头像：
 - button 组件的 open-type 设置为 **chooseAvatar**
 - 监听 **chooseavatar** 事件获取用户头像
 - 存储用户头像

注意：将调试基础库调整到最新



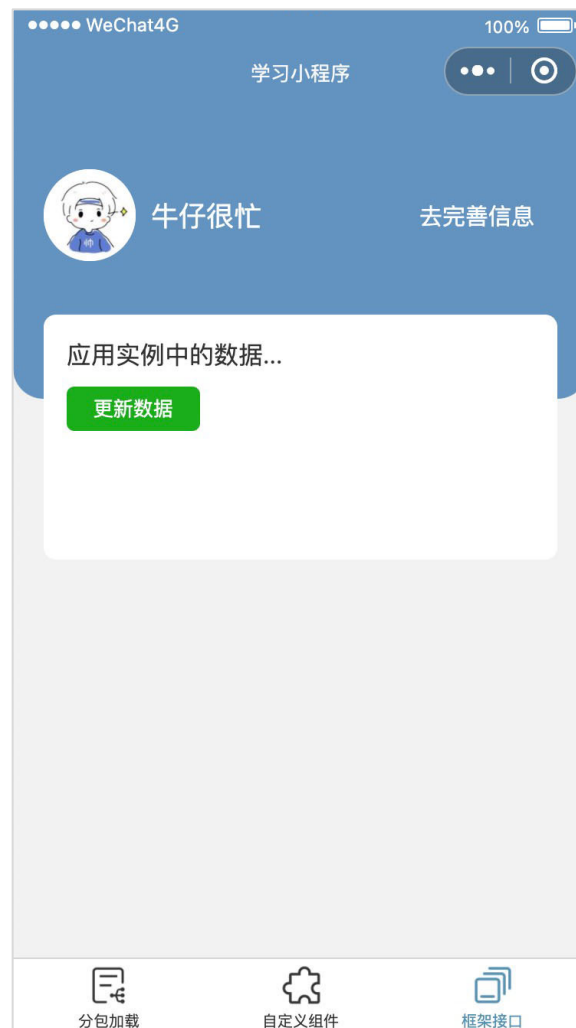
小程序框架接口：应用实例（数据共享）

- 基本用法：
 - App 函数中定义属性和方法
 - getApp 函数获取应用实例
- 应用示例：
 - onLaunch 中读取本地数据
 - 应用实例读取或更新数据



小程序框架接口：页面栈（页面实例）

- 基本用法：
 - getCurrentPages 获取当前页面栈（数组）
 - wx.redirectTo 关闭当前页，再跳转到新页面
 - wx.navigateTo 保留当前页，再跳转到新页面
- 页面实例：
 - data 页面初始数据
 - setData 更新数据
 - onShow 生命周期
 - route 页面路径





传智教育旗下高端IT教育品牌