MODALCUBE · 微信 AI 生态

微信 AI 接入实战手册

小程序接入「小微」的工程脚手架 —— 打开就能抄

不讲「小微是什么」，只讲**「怎么把它接进你的小程序」**。

全程提炼自官方文档 + WeStoreCafe demo 源码，**代码模板、决策树、避坑解法，照着抄**。

v1.0 · 整理于 2026-06 · 功能处官方内测，以官方最新文档为准

模酷科技 ModalCube 出品 · 内容引擎 **Gavin实战**

接入咨询 modalcube.com · 关注「Gavin实战」获取最新版

START 怎么用这本手册

按你的角色，直奔对应章节，不必通读。

你是谁	重点看	能拿到什么
■ **老板 / 决策者**	PART 0（5 分钟决策）	该不该接、选哪种模式、要投入多少、自己做还是找人做
◆ **开发者（要动手）**	PART 1 → 2 → 3 → 4 全程	路线图 + 可抄代码模板 + 避坑解法 + 调试评测上线
▸ **已上手，随手查**	附录 · 核心速查表	三层架构 / 字段 / 场景值 / 红线 一页查

图例

- √ 正确写法 / × 错误写法 —— 关键处都给正反对照
- ✦ **铁律** —— 来自官方最佳实践、不可破的硬规则
- ▲ **红线** —— 内测期限制、提审前必查
- 出处 xxx —— 标明该内容提炼自哪个官方文档 / demo 文件，可验证

◆ **关于时机**：开发模式仍在内测，资格在路上。但脚手架可以现在就备好——模板先备进项目，拿到资格当天就能直接开干。

PART 0 5 分钟决策：该不该接、怎么接

给老板 / 决策者。不写代码，只回答四个问题：要不要接、接哪种、投入多少、谁来做。

① 你的业务该不该接微信 AI?

```
你的业务里，有没有「能被一句话调起」的服务流程?
（点单 / 订票 / 查询 / 预约 / 报修 / 下单 …）
|
├── 有，且你的客户本来就在微信里活动
|       └──► 【√ 值得接】用开发模式做差异化对话体验，抢内测窗口
|
├── 有，但想先低成本卡位、暂不投入开发
|       └──► 【● 先占坑】用自动模式，几乎不改代码就先被小微调起
|
└── 纯展示 / 无交易流程（如企业官网类）
        └──► 【○ 先观望】囤认知，等场景成熟再动手
```

② 自动模式 vs 开发模式，选哪个?

	自动模式	开发模式
改造成本	几乎不用改代码	需手写 SKILL（mcp.json + 接口 + 卡片）
体验	基础——能被调起、基本展示	深度定制对话流，可对话内下单支付
适合	先卡位、占坑、低成本试水	把"对话内完成交易"做成差异化卖点
现状	已有美团/京东/携程/得物等接入	内测中，暂不可提审正式版

建议：**先上自动模式占坑，同步用开发模式备好脚手架**，拿到内测资格即切换升级。

③ 接入大概要投入什么?

官方未公布标准工时，这里给**「要做哪几件事、难点在哪」**，不是精确报价。

要做的事	难点 / 卡点	谁能扛
申请开发模式资格	内测审核，需等待	有公众号主体即可
把业务拆成 SKILL / 原子接口	★ 拆分粒度、调用准确率，最吃经验	懂业务的开发
写接口 + 卡片 + 调提示词	★ AI 调用调优、防编造，需反复迭代	开发 + 懂提示词的人
评测（≥30 用例/SKILL）+ 真机	iOS 限定、评测工具配置	开发 + 测试

④ 自己做，还是找团队做?

自己做，如果你…

- 有懂小程序的开发
- 愿意花时间踩 AI 调优的坑
- 把它当长期能力来建设

找团队，如果你…

- 想快、想稳、想一次做对
- 缺 AI 提示词调优经验
- 要赶内测窗口期卡位

PART 1 从 0 到上线：接入全景路线图

先看这页建立全貌——你要经历哪几步、每步交付什么、卡点在哪。后面 PART 2-4 是每一步的细节。

每一步：做什么 · 交付物 · 卡点

步	做什么	交付物	主要卡点	详见
①	申请开发模式资格	公众平台开通权限	内测审核，需等待	PART 0
②	搭 SKILL 目录骨架	`app.json` 的 agent 声明 + 分包	记得开 lazyCodeLoading	PART 2.1
③	声明原子接口	`mcp.json`	description 决定调用准确率	PART 2.2
④	实现接口逻辑	`apis/*.js`	事实+动作、防编造、300s 超时	PART 2.3
⑤	绑卡片 + 写编排	组件 + `SKILL.md` / `AGENTS.md`	卡片 6 条限制、提示词权重	PART 2.4–2.7
⑥	真机 + 评测	评测用例 ⩾ 30/SKILL	仅 iOS、工具配置	PART 4
⑦	自查 → 提审	过自查清单	内测代码勿合入正式版	PART 4

▲ 全程三条红线（先记住，细节在各 PART）

- 真机仅 **iOS**，微信 ⩾ 8.0.74，基础库 ⩾ 3.16.1——别在安卓上浪费时间。
- 内测代码**不可提审、勿合入正式版**。
- 文字链只是**兜底**，核心流程硬塞会被降权。

PART 2 工程模板区 · 打开就能抄的脚手架

本区 7 块模板，按接入顺序排：**2.1 摆目录 → 2.2 声明接口 → 2.3 实现接口 → 2.4-2.5 写编排 → 2.6-2.7 做展示**。所有代码提炼自官方 WeStoreCafe demo，可直接套进你的业务。

2.1 一个 SKILL 长什么样：目录脚手架

动手第一步不是写代码，是**把目录摆对**。一个 SKILL = 一个完整业务场景的能力包，由下面几类文件组成。

① 目录结构（照着建）

```
your-miniprogram/
├── app.json                  // 在 agent 字段里声明你的 SKILL
├── AGENTS.md                 // 全局提示词，统管多个 SKILL      ≤10KB
└── skills/
    └── drink-skill/          // 一个 SKILL 一个目录（一个小程序 ≤ 30 个）
        ├── mcp.json          // 原子接口声明（给 AI 看的说明书）   ≤24KB
        ├── SKILL.md          // 业务编排（流程/注意事项）          ≤16KB
        ├── apis/             // 每个接口一个 JS 文件
        │   ├── searchDrinks.js
        │   └── confirmOrder.js
        ├── components/       // 原子组件（对话流里的卡片）
        │   └── recommended-drinks/
        └── data/  utils/     // 你的业务数据与工具函数
```

② app.json 里声明 SKILL（真实写法）

```
{
  "lazyCodeLoading": "requiredComponents",   // 必开：按需加载组件
  "subpackages": [
    { "root": "packageDetail", "pages": ["pages/sku-picker"] }  // 半屏页放分包
  ],
  "agent": {
    "skills": [
      { "rootPath": "skills/drink-skill", "skillId": "drink-skill" }
    ],
    "agentConfig": "AGENTS.md"            // 全局提示词文件
  }
}
```

↑ 提炼自 demo 根目录 `app.json` 出处 app.json

✦ 三个一开始就要做对的点

- **按"场景"拆 SKILL**：点单一个、售后另一个；一个小程序最多 30 个 SKILL。
- SKILL 放**独立分包**并开 `lazyCodeLoading`，别塞主包。
- SKILL 之间**不共享运行时状态**，各自独立。

2.2 mcp.json 黄金模板：写给 AI 看的"接口说明书"

mcp.json **不是给程序员看的 API 文档**，是给小微（会读自然语言的调度员）看的说明书。它决定小微**什么时候调哪个接口、参数从哪来**。

① 接口声明骨架（通用，可直接抄）

```
{
  "tools": [
    {
      "name": "searchDrinks",              // 语义化"动词+对象"，别用 func1
      "description": "根据关键词搜索饮品。当用户明确提到某饮品名称或类型时调用。",
      "inputSchema": {
        "type": "object",
        "properties": {
          "keyword": { "type": "string", "description": "用户想搜的关键词，如'拿铁''美式'" }
        },
        "required": ["keyword"]
      },
      "outputSchema": { /* 可选：计入长度，但能帮 AI 理解返回结构，建议写 */ }
    }
  ]
}
```

② description 决定调用准确率 —— 四条黄金写法

要点	× 错误	√ 正确（demo 真实写法）
写**业务语义**，不写技术实现	`"搜索"` / `"查数据库"`	"根据关键词搜索饮品。当用户明确提到某饮品名称或类型（如'拿铁''美式'）时调用"
标注**调用时机**	只说功能，不说何时调	"当用户没明确指定饮品、只是想喝点什么、需要推荐时调用"
ID 入参**声明来源+禁编造**	`"饮品id"`	"饮品唯一标识 id，**从搜索或推荐结果中获取，禁止编造**"
有顺序依赖**声明前置条件**	不写，AI 乱序调用	"调用前**必须已通过 confirmSku 确认规格**、通过 getAddress 获取地址"

↑ 四条全摘自 demo 真实 description 出处 mcp.json

③ 参数设计三原则

优先用 ID，不用自然语言	√ `drinkId:"d001"` × `drinkName:"拿铁"`（自然语言可能多匹配/表述有差异）
枚举值给全+说明含义	有限取值的字段写 `enum` 并逐项解释，别让 AI 猜
同义字段统一命名	同一含义在所有接口都叫 `drinkId`，别一处 `id` 一处 `drinkId`

▲ 长度红线

`mcp.json ≤ 24KB`（计长度时先除去所有 `outputSchema` 字段及空格换行）。接口多时把 description 写精炼，别灌水。

2.3 原子接口"事实 + 动作"四分支骨架

接口实现的核心**不是"返回数据"**，而是用 content 字段告诉 AI 三件事：**现在什么情况 · 下一步干什么 · 什么不能干**。

① 四个返回字段，先分清"给谁看"

字段	给谁看	写什么
`content`	**AI（权重最高）**	事实陈述 + 下一步动作建议，纯文本
`structuredContent`	AI + 卡片组件	结构化数据，给卡片渲染
`_meta`	**仅卡片（AI 看不见）**	图片地址等"渲染要、AI 不用懂"的数据
`isError`	渲染控制	`true` 时不渲染卡片

② 四分支骨架（任何业务接口都能套）

```
module.exports = async function (params) {
  const { keyword } = params || {};

  // 分支①  缺参/非法 → 陈述事实 + 让用户补什么 + 禁编造
  if (!keyword || typeof keyword !== 'string') {
    return { isError:false,
      content:'未提供关键词，请让用户说明想搜的饮品，不要编造。',
      structuredContent:{ drinks:[] } };
  }

  const drinks = search(keyword);

  // 分支②  空结果 → 事实 + 正确出口(换词/看推荐) + 禁令
  if (!drinks.length) {
    return { isError:false,
      content:`未找到"${keyword}"，建议换词或调 getRecommendedDrinks 看推荐，勿编造。`,
      structuredContent:{ drinks:[] } };
  }

  // 分支③  成功 → 事实(找到几款) + 下一步动作 + 绑卡片
  const list = drinks.map(d => ({ id:d.id, name:d.name, price:d.price }));
  return { isError:false,
    content:`已找到 ${list.length} 款，可让用户选一款看详情和规格。`,
    structuredContent:{ drinks:list },              // → 给卡片渲染
    _meta:{ ui:{ componentPath:'components/recommended-drinks/index' } } };  // → 绑组件
};

// 分支④  有顺序依赖的接口(如 confirmOrder) → 前置校验，缺哪步先补哪步
if (!drinkId || !skuOptions || !addressId) {
  return { isError:false,
    content:'订单信息不完整：1)选饮品 2)确认规格 3)选地址，缺哪步先补哪步。',
    structuredContent:{ ok:false } };
}
```

↑ 抽象自 demo 的 `searchDrinks.js`（①②③）与 `confirmOrder.js`（④），原样可跑 出处 apis/*.js

③ content 怎么写 ——"事实 + 动作"两段式对照

场景	× 只给数据	√ 事实 + 动作（AI 才知道下一步）
搜到结果	"找到了"	"已找到 3 款拿铁。**可让用户选一款查看详情和规格**"
空结果	"无结果"	"未找到相关饮品。**建议换词或浏览推荐**。不要编造"
下单成功	"成功"	"订单已支付，单号 ORDxxx，实付 28 元。**可告知用户 30 分钟内送达**"

✦ 三条铁律（来自官方最佳实践）

- ✦ `content` 与 `structuredContent` **不重复**：一个给 AI 决策、一个给卡片渲染。
- ✦ 每个失败分支都要给 **"正确出口"**，否则 AI 会瞎试、甚至编造。
- ✦ 硬约束在**代码里强校验**，别只写在 description 里指望 AI 自觉。

2.4 SKILL.md 业务编排模板

mcp.json 告诉 AI"每个接口能干嘛"，SKILL.md 告诉 AI"**这些接口该按什么流程串起来**"。

```
# 〔场景名〕SKILL          // 如：饮品点单 SKILL

## 你是谁 / 角色
你是一个〔奶茶店〕的〔点单〕助手，帮用户完成〔从浏览到下单支付〕的完整流程。

## 你能做什么            // 把接口能力翻译成"用户视角"
1. 推荐：用户不知道选什么时主动推荐
2. 搜索：按关键词找到对应项
3. 下单支付：确认订单并完成支付

## 业务流程              // ★最关键：把"意图→接口→操作→接口"画清楚
1. 用户表达需求      → 调 getRecommendedDrinks / searchDrinks
2. 用户选定          → 调 selectDrink 查看详情和规格
3. 确认规格          → 调 confirmSku 计算价格
4. 下单前取地址      → 调 getAddress（没有则引导 saveAddress）
5. 提交支付          → 调 confirmOrder

## 注意事项             // 全场景级的硬约束
- 所有 id 必须来自接口返回，禁止编造
- 下单前必须确认规格和地址；门店未营业/超配送范围，提前告知，别走到支付
- 支付前清晰展示：项目、规格、数量、总价、地址
```

↑ 提炼自 demo `SKILL.md` 出处 SKILL.md

✦ 写"业务流程"的诀窍

- 用 **"用户意图 → 调哪个接口 → 用户操作 → 再调哪个"** 的链路描述，别只罗列接口。
- 把 **"什么时候不能往下走"** 写明（如门店未营业 / 超配送范围，就别走到支付）。

2.5 AGENTS.md 全局提示词模板

一个小程序可能有多个 SKILL（点单、售后、会员…）。AGENTS.md 站在**所有 SKILL 之上**：统一人设、统一风格、跨场景红线。上限 10KB。

```
# 全局助手设定

## 总人设与风格
你是〔品牌名〕的智能助手，语气简洁友好，不啰嗦、不卖弄。

## SKILL 分工            // 让 AI 知道什么需求进哪个 SKILL
- 点单 / 购买   → drink-skill
- 退款 / 售后   → aftersale-skill
- 会员 / 积分   → member-skill

## 全局红线（所有 SKILL 通用）
- 所有 id / 订单号必须来自接口返回，禁止编造
- 涉及支付 / 退款，必须向用户复述关键信息并等确认
- 不确定就引导用户澄清，不要猜
```

✦ 提示词放哪最有效（注意力权重）

- 约束按权重**从高到低**放：接口 `description` > `SKILL.md` 流程 > `AGENTS.md` 全局。
- **一处声明原则**：同一条约束只写一处，多处重复反而降低 AI 重视度。
- **正向引导优先**：多写"该做什么"，每条禁令都配一个替代出口。

2.6 原子组件（卡片）骨架

原子组件把接口数据渲染成对话流里的卡片。它限制多，且"点了卡片后怎么回到对话"官方没明讲，**得拆 demo 才知道**。

① 最小卡片骨架（wxml + js）

```
<!-- index.wxml: 列表卡片 -->
<view wx:for="{{drinks}}" wx:key="id"
      bindtap="onTapDrink" data-id="{{item.id}}">
  <image src="{{item.imageUrl}}" mode="aspectFill" />
  <text>{{item.name}}</text> <text>¥{{item.price}}</text>
</view>
```

```
// index.js: 把"用户点了哪个"回传对话流 ★关键
Component({
  properties: { drinks: { type: Array, value: [] } },
  methods: {
    onTapDrink(e) {
      const { id } = e.currentTarget.dataset;
      // triggerEvent 通知小微: 用户选了某款 → AI 接着调下一个接口
      this.triggerEvent('drinkselect', { drinkId: id });
    }
  }
});
```

↑ 提炼自 demo `components/recommended-drinks/` 出处 components

② 卡片 6 条硬限制（违反就过不了）

维度	限制
网络 / 定时器	默认 × 不可（要用得声明"实时动态组件"单独过审）
事件	仅 `tap` + image 的 `load / error`
滚动	禁上下滚动，scroll-view 仅横向
尺寸	宽高比 4:1 ~ 1:1，初始化即定死
image	仅网络地址，仅 png / jpg
登录 / 支付	× 不可（放原子接口里做）

✦ 一句话记住

- 卡片只管**展示 + 把点击 triggerEvent 回传**；所有"干活"（联网 / 支付 / 校验）都放原子接口里。

2.7 半屏页面：卡片放不下的复杂交互

选规格、填地址这种**要表单输入**的，卡片承载不了，用半屏页面。它是卡片的延伸，**不允许跳转到其他页面**。

要点	说明
用在哪	选商品规格(SKU)、填 / 改收货地址、其他需表单输入的场景
放哪	放**分包**里（见 2.1 的 `subpackages`）
怎么识别	通过场景值 `1433 / 1434` 判断是半屏打开
红线	不允许跳转其他页面；关闭后返回对话流

▲ 场景值速查（判断入口来源）

- 卡片右上角进小程序：`1442 / 1443`
- 半屏页面打开：`1433 / 1434`
- 文字链拉起：`1435 / 1436`

PART 3 避坑表：根因 + 解法

官方文档分散在各处的"坑"，集中成一张表。每条给 **根因** + ×**错误** + √**正确**。

① 架构 / 数据流（最容易踩）

坑	根因	× → √
卡片里发网络请求	原子组件默认不能联网	× 组件 js 里 wx.request → √ 数据在接口取好，用 structuredContent 传给卡片
`content` 和 `structuredContent` 装反	没分清"给 AI"还是"给卡片"	× 把结构化数据塞 content → √ content 写事实+动作给 AI，结构化数据给卡片
渲染数据被 AI 看见	该放 `_meta` 的放了 content	× 图片地址写进 content → √ 纯渲染用数据放 _meta，AI 不可见
SKILL 之间共享状态	SKILL 运行时相互独立	× 指望 A SKILL 读 B 的内存 → √ 跨场景数据走接口/storage

② AI 调用准确率

坑	根因	× → √
AI 用自然语言传参	没强制用 ID	× 入参 drinkName:"拿铁" → √ 入参 drinkId，description 标"从结果取、禁编造"
AI 编造 id / 数据	没声明来源 + 没禁止	× "饮品id" → √ "从搜索/推荐结果获取，禁止编造"
AI 乱序调接口	没声明前置条件	× confirmOrder 不写依赖 → √ "调用前必须已 confirmSku、getAddress"
约束重复写反而失效	多处重复降低权重	× 同条约束写三处 → √ 只在权重最高处写一次
失败时 AI 瞎试	失败分支没给出口	× 只返回"失败" → √ 事实+正确出口+禁止动作 三件套

③ 环境 / 合规（提审前必查）

坑	根因	× → √
在安卓上调试	真机仅支持 iOS	× 安卓真机预览 → √ iOS + 微信 ⩾8.0.74 + 库 ⩾3.16.1
核心流程塞文字链	文字链是兜底，硬塞降权	× 主流程靠文字链跳转 → √ 核心流程对话内闭环，文字链只兜底
内测代码进正式版	内测代码不可提审	× 合入正式版提审 → √ 提审前移除，用自己的 appid
接口耗时超限	单次+中间件链共享 300s	× 接口里跑长任务 → √ 控制耗时，长任务异步化

PART 4 调试 · 评测 · 上线自查

代码写完不等于能上线。这一段是**从本地跑通到提审**的最后一公里。

① 开发者工具配置（先配对，否则白忙）

项	设置
开发者工具版本	Nightly Electron Build 最新版
编译模式	切到「小程序 AI 编译」
调试基础库	`3.16.1`
真机预览	iOS 设备、微信 ≥ `8.0.74`
评测 / 校验前	设置 → 安全设置 → **开启服务端口**

② 评测：用 `wxa-skills-eval` 批量验准确率

AI 调用对不对，不能靠手点几下感觉，要**批量跑评测**用数据说话。

- 每个 SKILL ≥ **30 条用例**，覆盖正常流程 / 边界情况 / 异常输入。
- 硬约束尽量做**代码强校验**，而非只写在提示词里——提示词会被概率绕过，代码不会。
- 「下单 / 支付 / 发票」类用**测试号**，并关闭免密支付。

③ 上线前自查清单（逐项打勾）

资格与环境

- ☐ 已申请通过「开发模式」 ☐ 工具/基础库/真机环境如上配好

架构与接口

- ☐ 按场景拆 SKILL（≤30），放独立分包开 lazyCodeLoading
- ☐ 接口单一职责、语义化命名；description 标了调用时机 + 严禁场景
- ☐ 入参优先 ID，ID 字段声明来源 + 禁编造；同义字段统一命名
- ☐ content 用"事实+动作"；与 structuredContent 不重复；纯渲染数据放 _meta
- ☐ 每个失败/空结果分支都给了"正确出口"

组件与合规

- ☐ 卡片守住 6 条限制（宽高比/事件/滚动/网络/image/支付）；点击用 triggerEvent 回传
- ☐ 半屏页放分包、不跳转；核心流程对话内闭环，文字链只兜底
- ☐ 内测代码未合入正式版，用自己的 appid；敏感信息脱敏

调优与评测

- ☐ 硬约束代码强校验；同一约束只写一处（放权重最高处）
- ☐ 每个 SKILL ≥30 用例跑过评测；支付类用测试号

附录 核心速查表（打印 / 截图随手查）

把全书最常查的硬信息压成一页。功能处内测，以官方最新文档为准。

◆ 三层架构

C 端	小微——用户的 AI 助手（14.32 亿月活入口）
开发层	AI 开发模式——把能力封装成 SKILL
交易层	微信支付 AI 专属卡——对话内下单支付

◆ 核心概念

`SKILL`	一个场景能力包（≤30/小程序）
`原子接口`	最小执行单元，独立 JS 环境
`原子组件`	把数据渲染成对话流卡片
`半屏页面`	卡片延伸（选规格/填地址）
`知识库`	RAG 兜底检索
`AGENTS.md`	全局提示词，统管多 SKILL

◆ 文件大小上限

`AGENTS.md`	10KB	`SKILL.md`	16KB
`mcp.json`	24KB*	`page-meta.json`	8KB

* 计长度时先除去 outputSchema 与空格换行

◆ 接口 vs 组件

	接口	组件
网络	√	×默认
登录/支付	√	×
定时器	√	×默认
storage	√	√
事件	—	tap+image
滚动	—	仅横向

◆ 接口返回值

`content`	给 LLM ★ 事实+动作
`structuredContent`	LLM + 卡片，结构化
`_meta`	仅卡片，LLM 不可见
`isError`	true 不渲染卡片

▲ 场景值

卡片进小程序	`1442/1443`
半屏页面打开	`1433/1434`
文字链拉起	`1435/1436`

■ 红线清单

- ✦ 内测代码不可提审、勿合入正式版
- ✦ 真机仅 iOS，微信≥8.0.74，库≥3.16.1
- ✦ 文字链是兜底，硬塞被降权
- ✦ 卡片默认不能联网
- ✦ 小微只读不动，高危动作需确认
- ✦ 支付/退款向用户复述并等确认

NEXT 看完之后，下一步

这本手册会随官方内测进展**持续更新**，关注获取最新版。

你想要	找谁 / 怎么做
◆ **系统学接入**	关注订阅号 **Gavin实战**，看「微信 AI 接入实战」连载——每篇深讲一个点，配踩坑过程
◆ **技术落地** (交给团队做)	找 **模酷科技 ModalCube**：项目开发 · 小程序开发 · AI 接入 · 技术咨询 · 软件代开发。微信 AI 接入 / SKILL 封装是我们的拿手方向之一
▸ **对接方式**	官网 modalcube.com/contact 留资（48h 回复）｜公众号后台回复「合作」

◆ **手册与连载的关系**：这本手册是**「工具书」**（模板、表格，查着用）；公众号连载是**「读物」**（故事、踩坑、单点深讲，读着懂来龙去脉）。手册给你「正确答案照抄」，文章讲「为什么这么做」——配合着用最快。

模酷科技 · 服务号
接入咨询 / 商务对接

Gavin实战 · 订阅号
看接入连载 / 领最新版