

AI 玩具专利生成工作流 · 编排逻辑

项目：20260518-ai-toy-patent-workflow 分支：master @ e519627 文档生成：2026-05-23
真源：仓库当前代码 + RULES.md

本文件是从源代码反向归纳的编排说明，不是规约。出现差异时以 src/lib/templates.ts 的 PACK_ORDER、PACK_TEMPLATES、VIDEO_TEMPLATES 以及 src/app/api/** 的路由实现为准。

0 · 目录

- 1. 顶层一图：4 阶段串行 + 平行视频
- 2. 数据真源与冻结版本
- 3. 阶段 A：输入 → 候选图
- 4. 阶段 B：九宫格选中
- 5. 阶段 C：角色锁定 (CharacterSpec + L1)
- 6. 阶段 D：四个图片包串行
- 7. 阶段 E：文案模板 (18 条)
- 8. 阶段 F：视频任务 (Seedance, 5 条)
- 9. 横切：持久化、审计、鉴权、轮询
- 10. 编排约束与"规约 vs 实现"差异
- 11. 已落地导出 / 未落地路线

1 · 顶层一图：4 阶段串行 + 平行视频

整个工作流是一条带 gate 的状态机，一个 GenSession 串起所有阶段的产物。横向四个图片包严格串行，包内单图4 并发 + 拓扑排序，文案 / 视频在 characterSpec 锁定后即可触发，但前端按"四包完成后"再开"做 UX 引导。



一句话总结

选中图 (L0) → 净化为 L1 → 用 L1 作为根锚图生成各包根模板 (L2) → 包内其它模板基于 L2 派生 (L3) → 全程通过 GPT image edit 而不是文本拼 URL，保证角色一致。

2 · 数据真源与冻结版本

符号	代码位置	值 / 含义
PACK_ORDER	src/lib/templates.ts:13	['patent', 'accessories', 'production', 'marketing'] — gate 校验唯一来源
PACK_LABELS	src/lib/templates.ts:6	patent=专利包 / accessories=配件包 / production=生产打样包 / marketing=宣发包
TEMPLATE_FREEZE_VERSION	src/lib/templates.ts:4	toy-pack-templates-v01 — 写入每个 ToyAsset.meta 和 ExportManifest

符号	代码位置	值 / 含义
FILENAME_SCHEMA	src/lib/templates.ts:3	{sessionId}_{characterSlug}_{pack}_{view}_{version}.{ext}
PACK_TEMPLATES	src/lib/templates.ts:1094	4 个包各自的模板数组，每个包指定根模板（其它模板的 anchorTemplateId 全部指向根）
PACK_ASSET_CONCURRENCY	src/lib/packGenerator.ts:155	4 — 包内单图并发上限
VIDEO_TEMPLATES	src/lib/templates.ts:15	5 条：旋转 / 开箱 / 触感 / 角色故事 / 工厂预览
TEXT_TEMPLATES	src/lib/templates.ts:106	18 条：项目 / 专利 / 生产 / 配件 / 宣发 / 视频脚本

各包模板规模与根锚

包	kind	根模板 (L2 锚)	模板总数	必需	可选
专利包	patent	patent_front	12	7	5
配件包	accessories	acc_inventory_sheet	13	12	1
生产打样包	production	prod_front_spec	19	15	4
宣发包	marketing	mkt_white_front	22	11	11

规模来源 `PACK_TEMPLATE_SUMMARY` (`src/lib/templates.ts:1101`)。宣发包末尾 5 条 `video_*` 是分镜板 (图片)，与 `VIDEO_TEMPLATES` 的真实视频任务同名但不同源。

3 · 阶段 A：输入 → 候选图

3.1 三种输入模式 (`ProjectInputMode`)

模式	API	九宫格生成	L0 是什么	角色锁定
idea 想法	POST /api/generate	GPT images/generations × N (4/8/12)，ref 图作为文本提示拼接	用户从九宫格选中的图	用户手动点 /api/character/lock，normal 净化
remix 二创	POST /api/projects/from-upload	GPT images/edits 基于上传图 × N，强制"原创化"提示	用户从九宫格选中的图	同 idea
replicate 复刻	POST /api/projects/from-upload	跳过，上传图直接作为 L0 selected	上传的主体图	自动调 buildCharacterSpec + strict 净化

模式	API	九宫格生成	LO 是什么	角色锁定
exte nd <div>扩展</div>	POST /api/projects/from-upload	同 replicate	同 replicate	同 replicate, 且把上传图按 role 预填到专利六视图槽位 (preFilledSlots)

3.2 上传 role → 专利槽位映射 (extend 模式)

src/app/api/projects/from-upload/route.ts:19

UploadedImageRole	映射到 AssetTemplate.id
view-front	patent_front
view-back	patent_back
view-left	patent_left
view-right	patent_right
view-top	patent_top
view-bottom	patent_bottom

命中预填槽的 pack asset 不会调 GPT，直接复用上传 URL (packGenerator.ts:326-356)。

3.3 Provider 选择

```
// src/lib/providers.ts:10
export function detectProvider(): Provider {
  return process.env.OPENAI_API_KEY ? 'gpt' : 'mock';
}
```

- **gpt**: 图片生图走 POST {GPT_API_BASE}/images/generations 或 /images/edits；文本结构化走 /responses + format: json_object
- **mock**: 返回 SVG 占位图 (笑脸 + 渐变背景)，仅用于跑通流程，不能生产用
- **视频不 mock**: Seedance 缺 Key 时直接 503

4 · 阶段 B: 九宫格选中

POST /api/select (src/app/api/select/route.ts) 支持 action: 'select' | 'reject' | 'reset'。select 时把图从 data/generated/ 复制到 data/selected/ 并把新 URL 写回 img.meta.selectedUrl。

前端键盘约定 (src/components/PromptPanel.tsx): 1-9 选中，Shift+1-9 打叉。被打叉的图保留可见，不会进入后续阶段，但仍在 audit DB 留痕。

5 · 阶段 C: 角色锁定 (CharacterSpec + L1 锚图)

5.1 两条路径

路径 1 — 普通锁定

POST /api/character/lock

1. 幂等：未 force 且当前 `spec.sourceImageId == imageId`，直接返回缓存
2. `buildCharacterSpec()`：调 GPT JSON 结构化输出
3. `cleanupCharacterAnchor()` 用 **normal** prompt 净化为白底
4. 写入
`characterSpec.cleanReferenceImageUrl`
= L1 锚图 URL

路径 2 — 上传/复刻锁定

POST /api/character/lock-from-upload 或
from-upload 自动触发

1. 有 userHint 时覆盖 `session.prompt`
2. `buildCharacterSpec()` 在
replicate/extend/upload 分支走
`inferCharacterSpecFromImage()`
(Vision 推断)
3. `cleanupCharacterAnchor()` 用 **strict**
prompt：仅抽取最大最完整的单一主角
色，丢弃多宫格 / 包装 / 海报版式
4. 强制 `force=true`，每次都重算并覆盖 L1

5.2 CharacterSpec 字段（src/lib/types.ts:76）

15 个语义字段 + 3 个图像引用 + lockedAt。详见 `CHARACTER_SPEC_FIELDS`（`templates.ts:58`）。关键三项：

- `sourceImageId / sourceImageUrl` — L0（用户选中或上传的图）
- `cleanReferenceImageUrl` — L1（净化后的白底锚图，是后续所有 pack 生成的根锚）
- `negativePrompt` — 写入每张 pack 图的 prompt 后缀，防角色漂移

5.3 strict 净化的关键约束（节选）

src/lib/packGenerator.ts:171-200

- 多宫格 / 品牌手册 / 包装展示 → 只抽取最大最清楚的单一主角色，不保留版式 / 分割线 / 标题 / 包装平铺
- 必须保留：玩具本体的设计标识、衣服图案、帽标、面罩声波图案等用户上传的原创品牌符号
- 背景纯白，去水印 / 价格 / 网页 UI
- 不改五官、配色、配件位置、材质纹理

6 · 阶段 D：四个图片包串行

6.1 三道 gate

每次 POST /api/packs/generate 前后端都过的 gate

1. `characterSpec` 必须存在 — 否则 409 "请先锁定角色设定" (`packs/generate/route.ts:43`)
2. 前一包必须 `complete` — `PACK_ORDER` 中前一项必须满足 `pack.status === 'complete'` 且模板覆盖率 100% (`packs/generate/route.ts:25-58`)
3. 并发互斥 — 同一 `session:image:kind` 已在跑则返回 202 running (`generationLocks.ts`)
4. 额外约束: 源图 `status` 必须 = `selected`

6.2 包内编排 (`generateAssetPack` , `packGenerator.ts:276`)

```
sortTemplatesByAnchor(getPackTemplates(kind)) // 拓扑排序
↓
取/建 CharacterSpec → cleanupCharacterAnchor // 兜底确保 L1 存在
↓
existingPack 合并: 从断点续生 (按 templateId 去重)
↓
takeReadyTemplate() // 依赖已就绪的模板进入候选
↓
inFlight ≤ PACK_ASSET_CONCURRENCY (=4) // 并发槽
↓
对每张模板:
· 若命中 preFilledSlot → 直接复用上传图, 不调 GPT
· 否则 generateAssetImage():
  · anchorImageUrl = anchorAsset.url // L3: 基于已生成根模板
    ?? L1.cleanReferenceImageUrl // L2: 用净化锚图
    ?? L0.url
  · GPT images/edits 真正的图生图 (读 anchor 字节 → multipart)
  · data: 开头则落盘到 data/packs/{packId}_{assetId}.{ext}
↓
async onProgress(pack) → persistPackProgress (每张都回写 session JSON)
↓
全部就绪后 pack.status = 'complete', 写 ExportManifest 到 data/exports/
```

6.3 派生层级 (`ToyAsset.derivationLevel`)

层	含义	来源 URL	触发条件
L0	用户选中 / 上传主体图	<code>img.url</code>	选中 / 复刻
L1	白底净化锚图	<code>characterSpec.cleanReferenceImageUrl</code>	角色锁定

层	含义	来源 URL	触发条件
L2	每个包的根模板图	<code>data/packs/...</code>	包内 <code>anchorTemplateId == undefined</code> 的模板（每包仅一张： <code>patent_front</code> / <code>acc_inventory_sheet</code> / <code>prod_front_spec</code> / <code>mkt_white_front</code> ）
L3	包内其它图	同上， <code>basedOn = L2</code>	所有 <code>anchorTemplateId</code> 指向根的模板

代码里 `derivationLevel` 只被赋值 `2`（无 `anchorAsset`）或 `3`（有 `anchorAsset`）。`0/1` 出现在类型定义中，运行时由 `L0` 图片本身和 `cleanReferenceImageUrl` 隐式承担。

6.4 单张重做（`POST /api/assets/[assetId]/regenerate`）

双重 gate

- `confirmCost === true` 才放行（前端必须二次确认），否则 `400`
- 同 `session:asset` 并发锁，已在跑返回 `429`
- 沿用同一 `anchor`：优先该 `asset` 的 `anchorAsset` → `cleanReferenceImageUrl` → `sourceImageUrl` → `L0`
- 支持 `userRefinement` 文本追加到 `prompt` 末尾

6.5 增量回写与断点续跑

`onProgress` 在每张生成完成后 `reload session JSON`、用最新 `pack` 替换旧版本（按 `kind + sourceImageId` 匹配），再写回。`generateAssetPack` 启动时会取出未完成的 `existingPack`，按已落地的 `templateId` 跳过、只生成剩余项 → 断网或失败可重试。

7 · 阶段 E：文案模板（18 条）

7.1 路由

`POST /api/text/generate`，body `{sessionId, templateIds?}`。后端唯一 `gate`：
`session.characterSpec` 必须存在（`text/generate/route.ts:18`），不强制四包完成。

7.2 实现

`src/lib/textGenerator.ts`

- 未传 `templateIds` 时生成全部 18 条；传了则只生成子集
- 一次 `GPT /responses JSON` 调用，要求返回 `{items: [{templateId, content}]}`
- 未配 `GPT Key` 时每条用 `fallbackContent()` 生成占位稿，标注“未配置文本模型时生成占位稿”
- 结果按 `templateId` 去重后写入 `session.textAssets[]`

7.3 18 条文案模板按 `kind` 分组

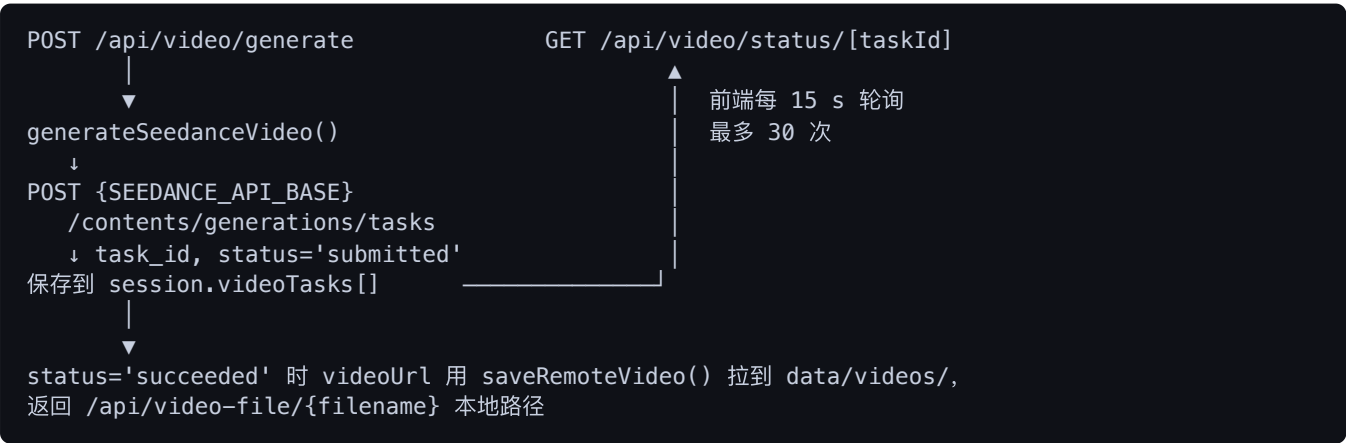
kind	条 数	典型 templateId (必需打 ★)
project	2	★ text_project_design_brief · ★ text_character_setting
patent	7	★ product_name · ★ product_use · ★ design_points · ★ representative_view · ★ view_brief · color_claim
production	4	★ brief · ★ cmf · ★ bom · ★ qc
accessories	2	★ accessory_brief · ★ accessory_bom
marketing	3	★ core_copy · ★ detail_page · social_posts
video	1	video_script_pack (脚本文字包)

8 · 阶段 F：视频任务 (Seedance)

8.1 五条视频模板 (VIDEO_TEMPLATES)

id	标题	比例	时长
video_turntable	360 度旋转展示	16:9	6 s
video_unboxing	开箱短片	9:16	8 s
video_touch_detail	触感细节	9:16	6 s
video_story_intro	角色故事介绍	16:9	8 s
video_factory_preview	工厂预览短片	16:9	8 s

8.2 提交 + 轮询



8.3 锚图优先级 (page.tsx:580–589)

- mkt_white_front — 宣发白底正面图 (最稳定)
- patent_front — 专利主视图

- 3. `characterSpec.cleanReferenceImageUrl` — L1 净化锚图
- 4. 当前选中意向图 L0

8.4 PUBLIC_APP_URL 注入

Seedance 需要从公网拉参考图，所以 `publicUrlOrUndefined()` 把 `/api/img/...` 用 `PUBLIC_APP_URL`（生产 = `https://ai-toy.kang-kang.com`）转成绝对 URL。localhost / 127.0.0.1 / 私有 IP 一律丢弃。

8.5 视频任务去重

每次新提交按 `templateId` 去重覆盖（`video/generate/route.ts:46`），保证 5 个模板各最多一个最新任务。fix: dedupe suffixed video tasks（7abbb7d）专门处理 `video_turntable_60s` 等带后缀的真实成片回流到默认模板卡。

9 · 横切：持久化、审计、鉴权、轮询

9.1 八个存储桶（`src/lib/storage.ts`）

桶	URL 前缀	放什么
<code>data/sessions/</code>	—	每个 session 一个 JSON，含 images / packs / textAssets / videoTasks / exports 全量
<code>data/generated/</code>	<code>/api/img/generated/</code>	九宫格候选图原图
<code>data/selected/</code>	<code>/api/img/selected/</code>	选中后复制一份（保留生成版本不被覆盖）
<code>data/refs/</code>	<code>/api/img/refs/</code>	idea 模式上传的参考图
<code>data/uploads/</code>	<code>/api/img/uploads/</code>	remix / replicate / extend 的上传图
<code>data/anchors/</code>	<code>/api/img/anchors/</code>	L1 净化锚图 <code>{sessionId}_{imageId}_clean.{ext}</code>
<code>data/packs/</code>	<code>/api/img/packs/</code>	四个包的所有 ToyAsset 图片
<code>data/videos/</code>	<code>/api/video-file/</code>	Seedance 成片从公网拉回的本地副本
<code>data/exports/</code>	<code>/api/export/</code>	ExportManifest JSON（每个 pack 一份）

9.2 审计：SQLite + 兜底 JSONL

`src/lib/auditDb.ts`。每个 API 路由的关键节点（started / completed / failed / blocked / saved）都调 `recordEvent()`，落到 `data/app.db`。Docker 镜像内置 `sqlite3`；非 Docker 本地缺 `sqlite3` 时降级写 `data/audit-fallback.jsonl`，不阻断流程。

每张图也通过 `upsertImageAsset()` 写入 `image_assets` 表, 包含 `bucket / width / height / sizeBytes / kind / templateId / origin`, 是 `/api/gallery/[sessionId]` 的真源。

9.3 鉴权 (`src/middleware.ts`)

- Cookie 名: `WEB_AUTH_COOKIE_NAME` (默认 `ai_toy_session`)
- HMAC-SHA256 签名 `body.signature`, `body` 是 base64url 编码的 `{u, exp}`
- 公开路径: `/login / /_next/ / /api/auth/ / /api/img/ / /favicon.ico / /robots.txt / /sitemap.xml`
- 未鉴权: HTML 路径 302 到 `/login?next=...`; 非 HTML API 返回 `401 {error: 'unauthorized'}`
- `/api/img/*` 故意保持公开 —— Seedance 必须能从公网拉参考图

9.4 轮询节奏 (前端)

对象	间隔	最大次数	终止条件
pack 生成 (<code>scheduleSessionRefresh</code>)	5 s	90	无 <code>status='draft'</code> 的 pack; 前 6 次无论如何都跑
视频任务 (<code>scheduleVideoRefresh</code>)	15 s	30	status 不再是 <code>submitted/processing</code>

10 · 编排约束与"规约 vs 实现"差异

差异 1: RULES.md 说"四个图片包完成后才解锁文案和视频"

后端实际只校验 `session.characterSpec` 存在:

- `/api/text/generate`: 只 check `characterSpec` (`text/generate/route.ts:18`)
- `/api/video/generate`: 完全不 check pack 完成度, 直接打 Seedance

这条规约靠前端 UX 引导执行, 不是后端 enforce。绕过前端可以在锁定角色后立刻发文案/视频请求。

差异 2: 视频不 mock

没配 `SEEDANCE_API_KEY` 时 `/api/video/generate` 和 `/api/video/status` 返回 `503`, 不会回退到占位视频。文档和 RULES.md 一致。

差异 3: 宣发包里 5 条 `video_*` 模板是分镜板 (图片), 不是真实视频

`marketing` 包模板列表里 `video_turntable / video_unboxing` 等 5 条 id 与 `VIDEO_TEMPLATES` 重名, 但 `kind=marketing`、`aspectRatio=16:9` 或 `9:16`, 走的是 GPT image edit, 产出 PNG 分镜板。真实视频由 Seedance 异步任务单独产出, 存 `session.videoTasks[]`。两者完全独立, 前端按 `templateId` 关联展示。

差异 4：派生层级运行时只用 2 / 3

类型定义 `derivationLevel: 0 | 1 | 2 | 3` 给出了完整四级，但 `generateAssetPack` 只赋值 2（包根模板）和 3（包内其它）。L0/L1 由 `GenImage` 和 `CharacterSpec.cleanReferenceImageUrl` 隐式承担，不写入 `ToyAsset.derivationLevel`。

差异 5：preFilledSlot 命中后 derivationLevel

命中预填上传图时仍按 `anchor` 存在与否赋 2/3（`packGenerator.ts:347`），但实际生成 URL 是上传桶 URL，不是 packs 桶。导出 ZIP 时 `extFromAsset` 会从 URL 抓扩展名，`readImageUrl` 回到 `uploads` 桶读字节。

11 · 已落地导出 / 未落地路线

11.1 已落地

- **ExportManifest JSON**：每包生成结束自动写 `data/exports/{sessionId}_{kind}_{version}_manifest.json`，含 `files[]`（`asset_id`, `templateId`, `filename`, `url`, `anchor`, `derivation`, `checklist`）
- **ZIP 下载**：GET `/api/packs/download?sessionId=&kind=`，纯 Node Buffer 拼装 ZIP（含 CRC32），文件名 `{characterSlug}_{kind}_{N}张.zip`，按 `templateId` 顺序编号 `01_xxx.png`

11.2 未落地（RULES.md 路线）

- **PNG 高清导出 + PDF 合订**：ExportManifest 已预留 `exportTargets: ['zip', 'pdf', 'manifest-json']`，只实现了 zip + manifest，pdf 未生成
- **Seedance 任务轮询 UI**：现状是被动 15s 间隔静默 refresh，没有进度条 / 失败重试按钮的完整 UI

12 · 关键 API 速查

方法	路径	gate / 关键行为
POST	<code>/api/uploads</code>	multipart, role 必传
POST	<code>/api/generate</code>	idea 模式批量生图 (4/8/12)
POST	<code>/api/projects/from-upload</code>	<code>mode</code> ∈ {remix, replicate, extend}, replicate/extend 自动锁定 strict
POST	<code>/api/select</code>	<code>action</code> ∈ {select, reject, reset}, select 时复制到 selected/
POST	<code>/api/character/lock</code>	普通净化；force 控制是否重算
POST	<code>/api/character/lock-from-upload</code>	strict 净化；force 总是 true
POST	<code>/api/character/cleanup</code>	独立触发 <code>cleanupCharacterAnchor</code>

方法	路径	gate / 关键行为
POST	/api/packs/generate	三道 gate; background=true 时返 202 异步跑
POST	/api/assets/[assetId]/regenerate	必传 confirmCost=true; 并发锁
GET	/api/packs/download?sessionId=&kind=	按选中图找该 kind 的 pack 打 ZIP
POST	/api/text/generate	必须 characterSpec; 可传 templateIds 子集
POST	/api/video/generate	必须 Seedance Key; 按 templateId 去重覆盖
GET	/api/video/status/[taskId]?sessionId=	查 Seedance + 写回本地副本
GET	/api/sessions	按 createdAt desc 列全部 session 元信息
GET	/api/templates	把 PACK_TEMPLATES / TEXT / VIDEO 暴露给前端
GET	/api/gallery/[sessionId]	从 image_assets 表 + filesystem 拼图库
GET	/api/audit/[sessionId]	读 events 表事件流水
GET	/api/img/[bucket]/[filename]	公开, Seedance 拉参考图依赖
GET	/api/video-file/[filename]	本地视频副本
POST	/api/auth/login / /logout	HMAC HttpOnly Cookie

附录 · 文件锚点

关键概念	代码位置
串行顺序 PACK_ORDER	src/lib/templates.ts:13
包模板冻结版本	src/lib/templates.ts:4
包内并发上限	src/lib/packGenerator.ts:155
包 gate 三道	src/app/api/packs/generate/route.ts:42-91
包内拓扑 + 并发调度	src/lib/packGenerator.ts:392-424
L1 strict / normal prompt	src/lib/packGenerator.ts:171-200
L1 净化路径	src/lib/packGenerator.ts:157
L0/L1/L2/L3 派生	src/lib/packGenerator.ts:316-389
preFilledSlot 映射	src/app/api/projects/from-upload/route.ts:19
视频锚图优先级	src/app/page.tsx:580-589
视频任务 templateId 去重	src/app/api/video/generate/route.ts:46
pack 进度轮询	src/app/page.tsx:536-543

关键概念	代码位置
video 状态轮询	<code>src/app/page.tsx:545-557</code>
generationLocks 全局并发锁	<code>src/lib/generationLocks.ts</code>
ZIP 打包	<code>src/app/api/packs/download/route.ts</code>
HMAC Cookie 鉴权	<code>src/middleware.ts</code>
审计写库	<code>src/lib/auditDb.ts</code>

— 文档生成基于 commit `e519627` 。结构性改动后请重跑 `npm run docs:orchestration` （如已配脚本）或重新执行 docs/orchestration.html 的生成命令。