基于SDD与Harness工程的
全流程AI项目实践

以戏单可视化服务平台为例

陈尚松 | 上海福呈数据科技有限公司 技术总监

2026-07-07

基于SDD与Harness工程的

全流程AI项目实践

交流目的

以真实项目《戏单可视化服务平台》为案例，从零到一完整实现

带大家走完以下过程：需求描述 → SDD设计 → Harness骨架 → Qoder编码 → 验收交付

目录

1

案例介绍

以《戏单可视化服务平台》为例

2

Vibe Coding开发理念及工具

SDD设计 → Harness骨架

→ Qoder编码

3

项目案例实践过程

从0到项目落地全过程

4

Vibe Coding实践复盘&进阶

真实踩坑案例

引入Loop Engineering后的开发过程

01

案例介绍

背景

项目背景

近年来，上海图书馆积极推进特色文献特藏建设，征集保存了数量丰富、年代跨度广的戏单馆藏。

戏单作为戏曲演出配套的原生宣传文献，详实记载着历代戏曲演出的剧目、演员、剧团、演出场次、剧场信息，沉淀着鲜明的地域民俗与大众审美特色。

它既是戏曲史学、表演艺术、近代文娱产业研究的核心一手史料，完整还原传统戏曲的发展脉络；同时串联起传统戏曲在不同社会阶段的传承、革新与传播轨迹，具象展现传统文艺伴随社会现代化发展的演变历程，是研究中国式现代化背景下传统艺术变迁不可多得的独特实物样本。

同时上海图书馆本次开放数据竞赛，开放800余种戏单数据供广大参赛者研究使用。

戏单可视化服务平台功能概览

智能检索

关键词检索 + 高级筛选

自然语言检索（AI解析）

检索词标签(chips)展示

列表展示

卡片式戏单展示

支持网格/列表切换

分页加载

详情查看

事件/载体/作品三个标签页

完整戏单信息展示

关联数据跳转

AI助手

自然语言问答

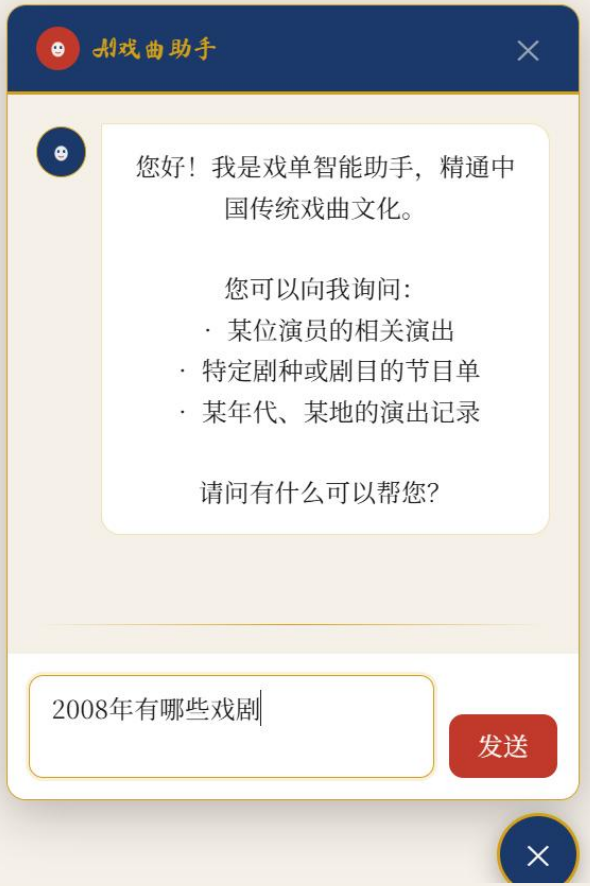
多轮对话记忆

快捷示例引导

戏单可视化服务平台主要功能

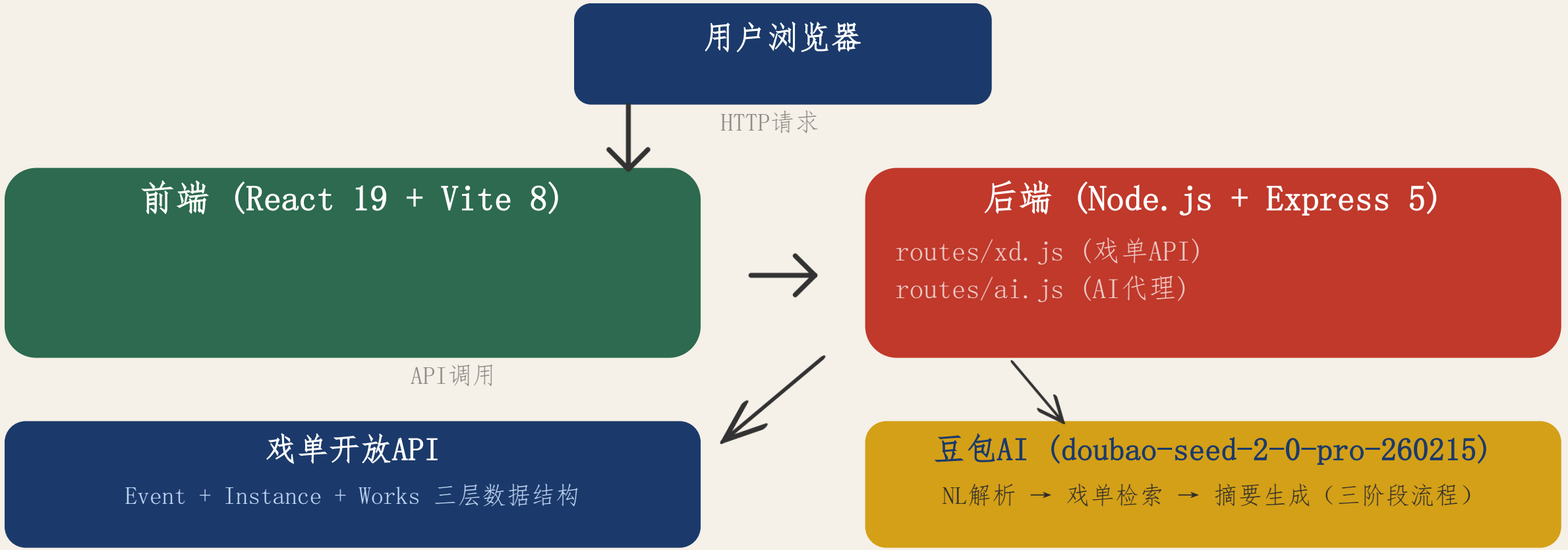
戏单可视化服务平台主要功能

戏单可视化服务平台主要功能



戏单可视化服务平台主要功能

戏单可视化服务平台数据流程图



安全层：CORS白名单 | Rate-limit 15次/IP/min | API Key不出后端 | .env密钥管理

戏单可视化服务平台项目架构图

前端组件架构

pages/	ListPage, DetailPage
components/Layout/	Header, Footer, MainLayout
components/Search/	SearchBar, AdvancedSearch
components/List/	ProgramCard, ProgramGrid
components/Detail/	EventTab, CarrierTab, WorksTab
components/AI/	ChatBot, ChatMessage
context/ + hooks/	SearchContext, usePrograms
services/	api.ts (统一API封装)

后端服务架构

routes/xd.js	戏单列表/详情API
routes/ai.js	AI代理（NL解析+聊天）
services/xdService.js	戏单数据服务（上海图书馆API）
services/doubaoService.js	豆包AI服务（OpenAI兼容）
middleware/	错误处理、日志、限流
.env	API Key等敏感配置

技术栈

前端：React 19 + Vite 8 + Tailwind CSS 3 + React Router 7

后端：Node.js + Express 5 + Axios

AI：豆包 doubao-seed-2-0-pro-260215

数据：戏单开放API

02

Vibe Coding开发理念及工具

AI工程四阶段演进

从"人写提示词"到"系统自主迭代"的范式跃迁

Prompt Engineering

2022-2024　　模型层

优化单轮文本输入
人工编写提示词
一次生成一次验收

人 → 操作者

▼

Context Engineering

2024-2025　　信息层

管理动态数据环境
RAG / Memory / 知识库
为模型组织上下文

人 → 策展人

▼

Harness Engineering

2025-2026　　系统层

沙箱隔离 + 工具集成
安全护栏 + 权限管控
每次执行可控可信

人 → 监督者

▼

Loop Engineering

2026+　　架构层

自主迭代闭环系统
Judge自动验收 + 自动修正
7×24无人值守运行

人 → 架构师

每一层叠加在前一层之上，不是替代关系。Loop工程需要Harness基础才能安全运行。

核心概念总览

SDD
Spec Driven Development

形式化的软件设计文档

涵盖架构、组件、API契约

数据模型、非功能约束、UI/UX

Harness
开发脚手架 + 约束框架

项目骨架、构建配置

运行时脚手架、编码规约

安全脚手架、测试脚手架

Qoder
AI代码编辑器

对话式编码、上下文感知

多工具协同、Memory机制

将SDD要素持久化为AI知识

Quest
需求驱动引擎

用户故事→结构化SDD要素

辅助梳理功能边界、接口设计

生成Qoder可消费的开发指令

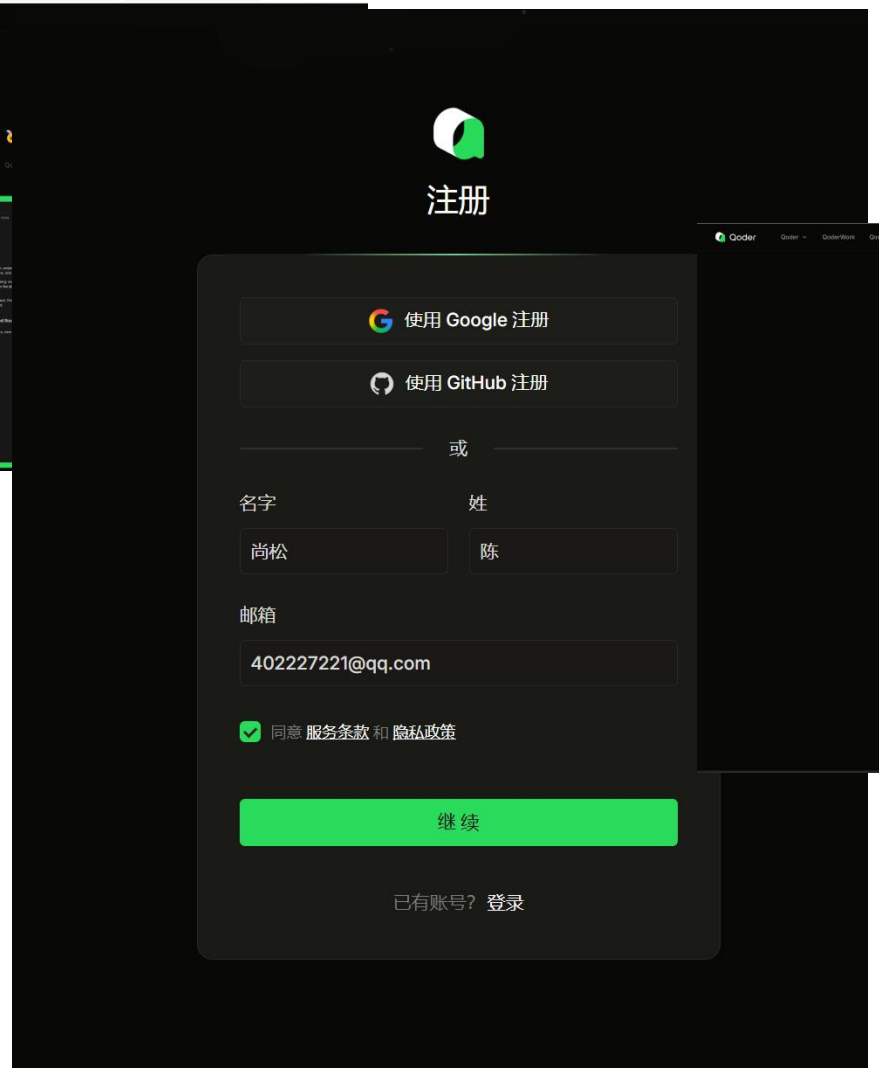
Loop
自主迭代开发引擎

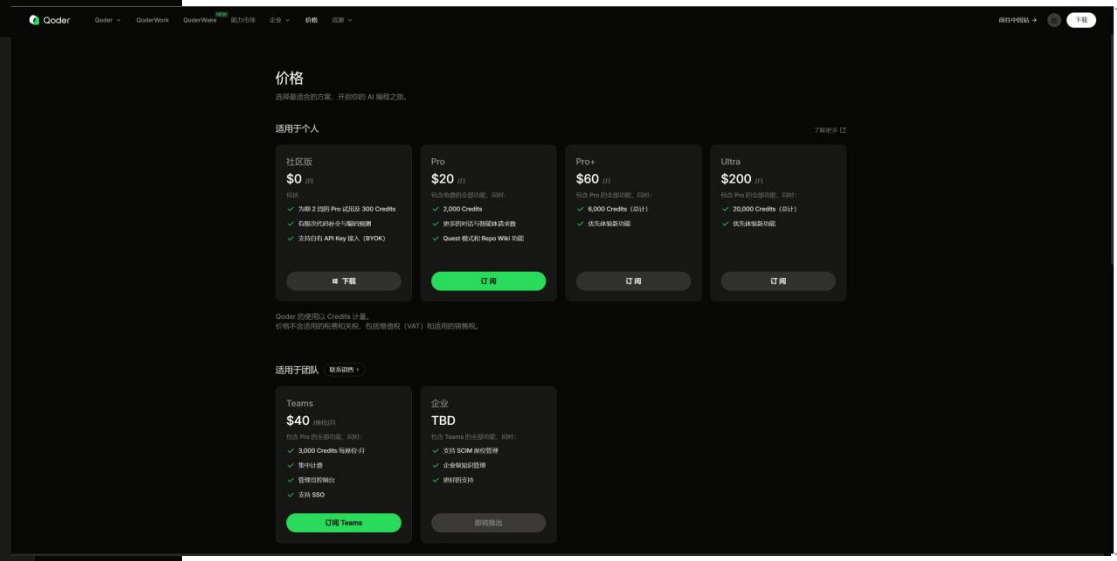
SDD→Qoder生成Judge→生成代码→自动验收迭代

Plan→Do→Check(Judge)→Act 闭环，不通过则自主修复

人设计规则，AI自主运转；从"人驱动AI"到"AI自主开发"

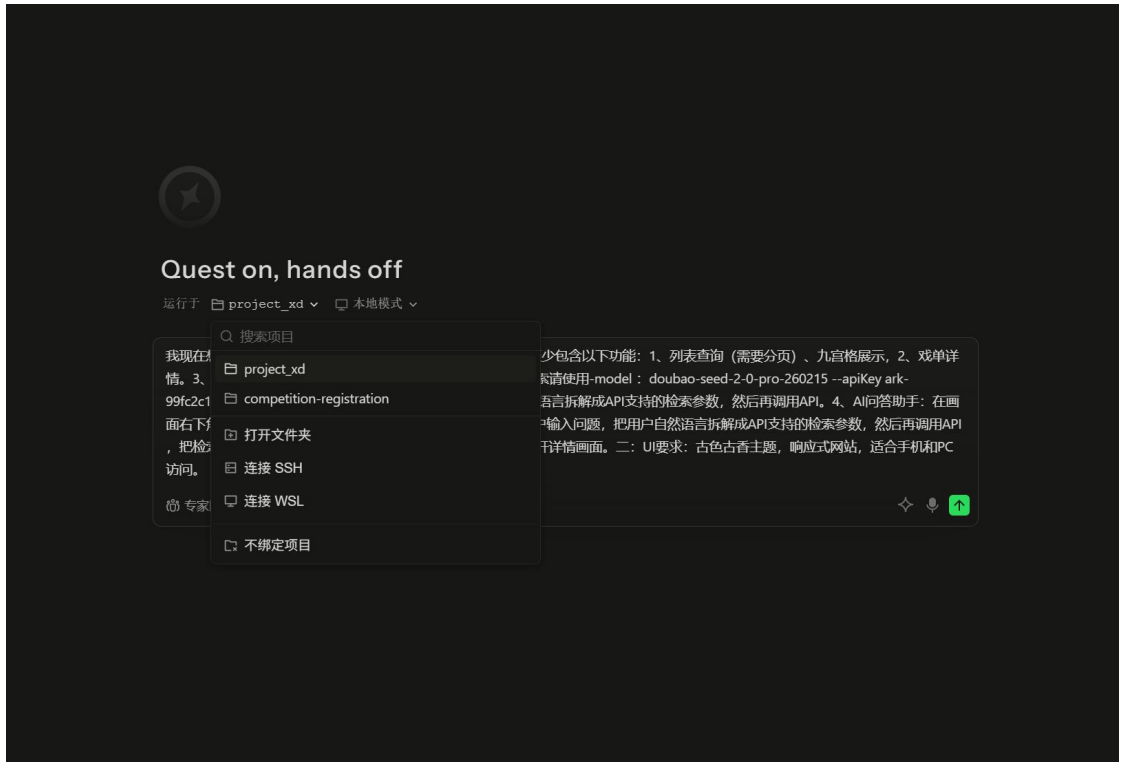
核心流程一：下载Qoder

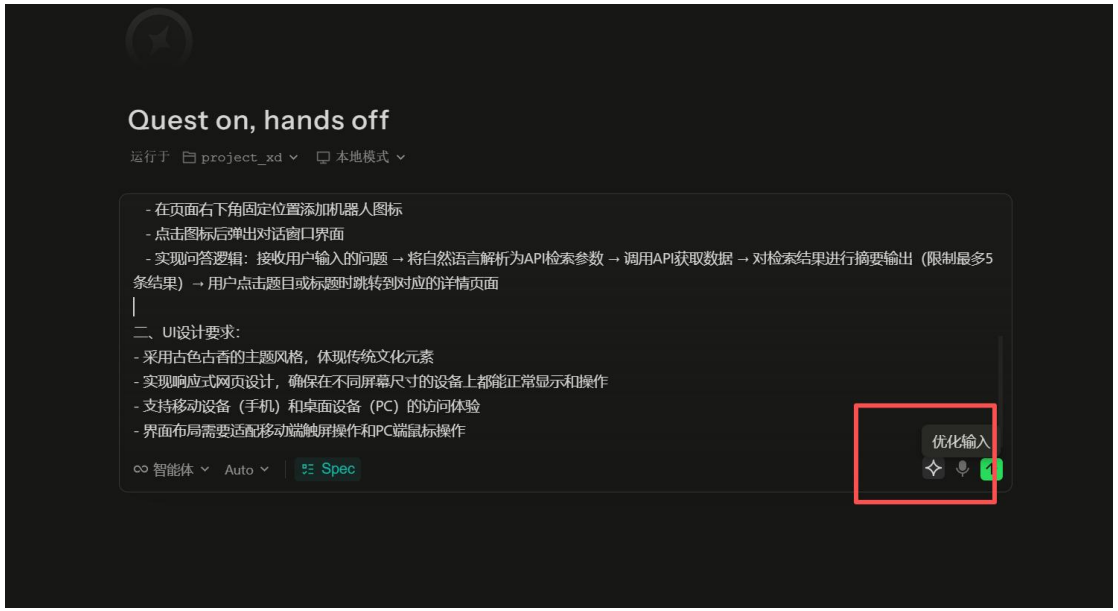




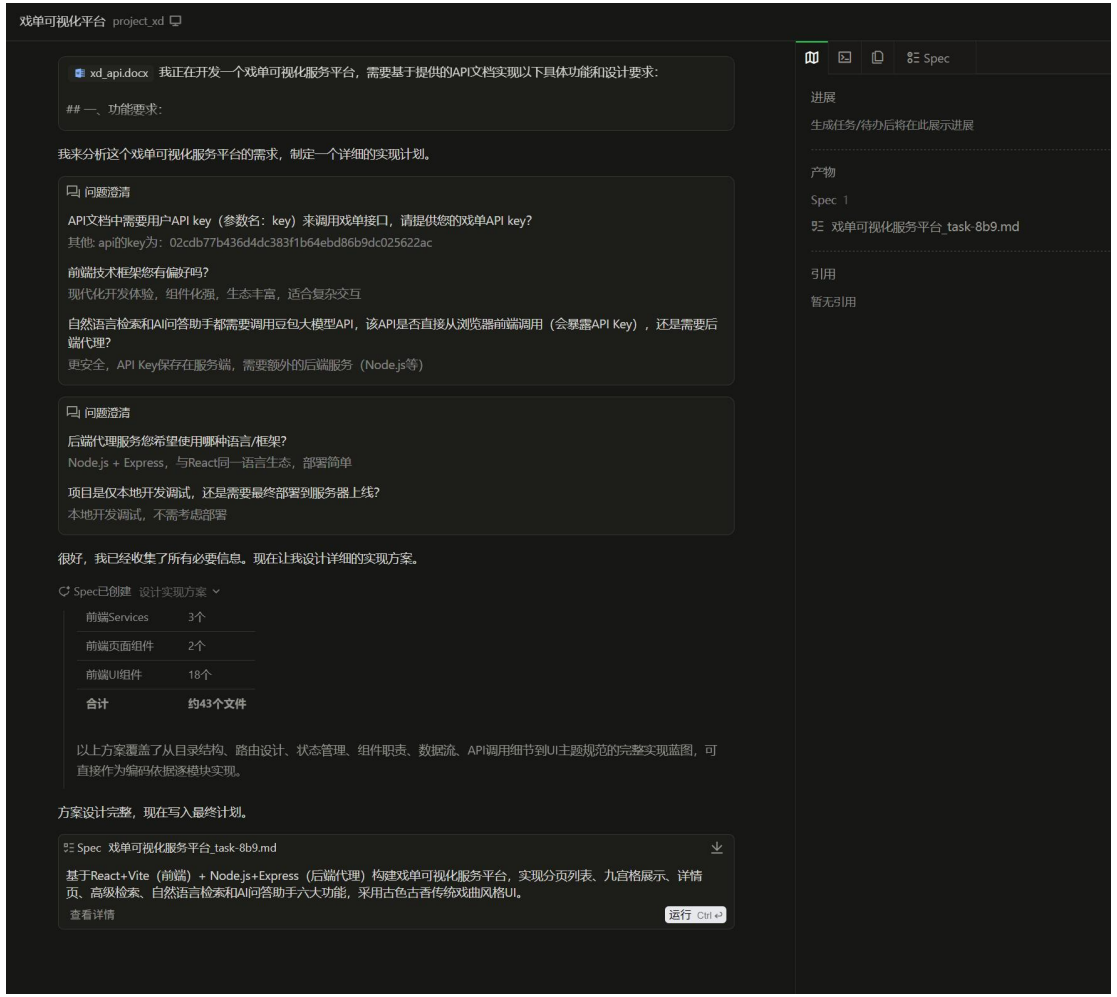
https://qoder.com/zh

核心流程二：Quest构建需求提示词

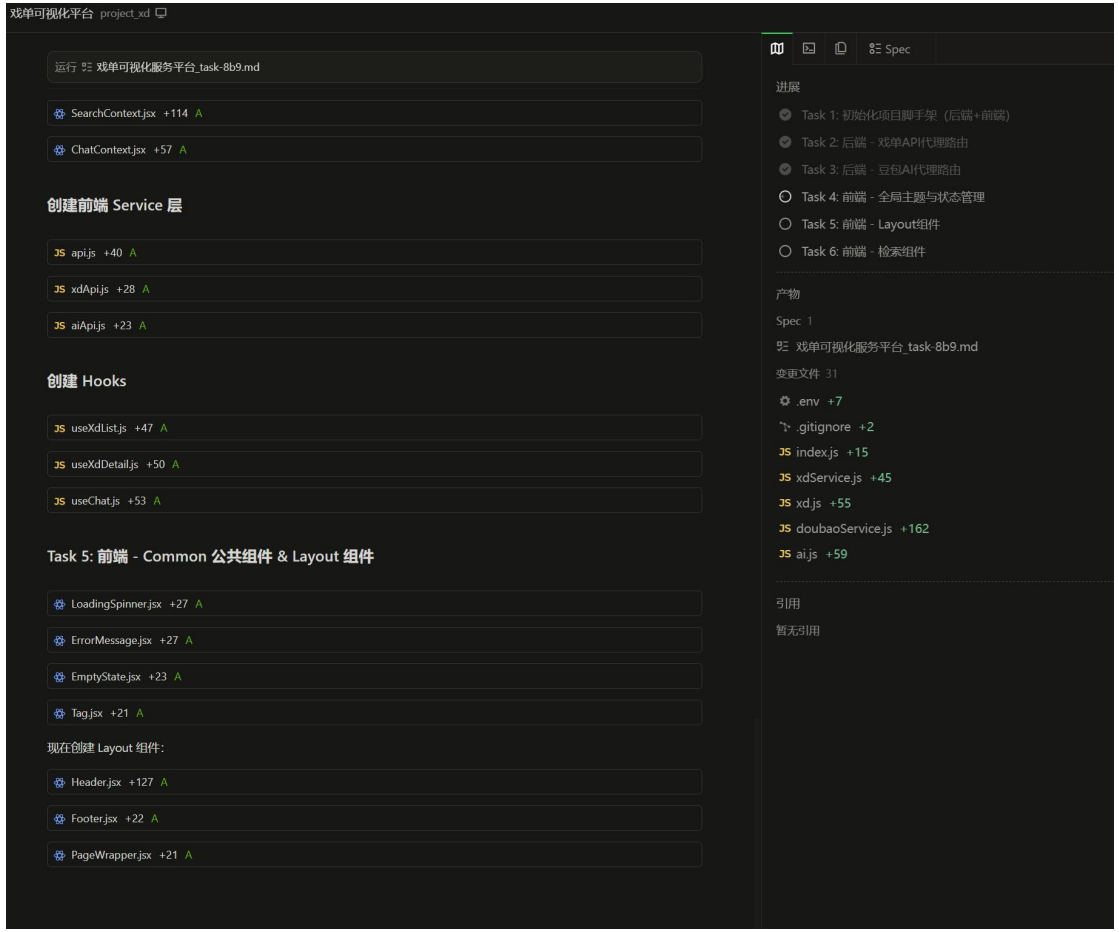


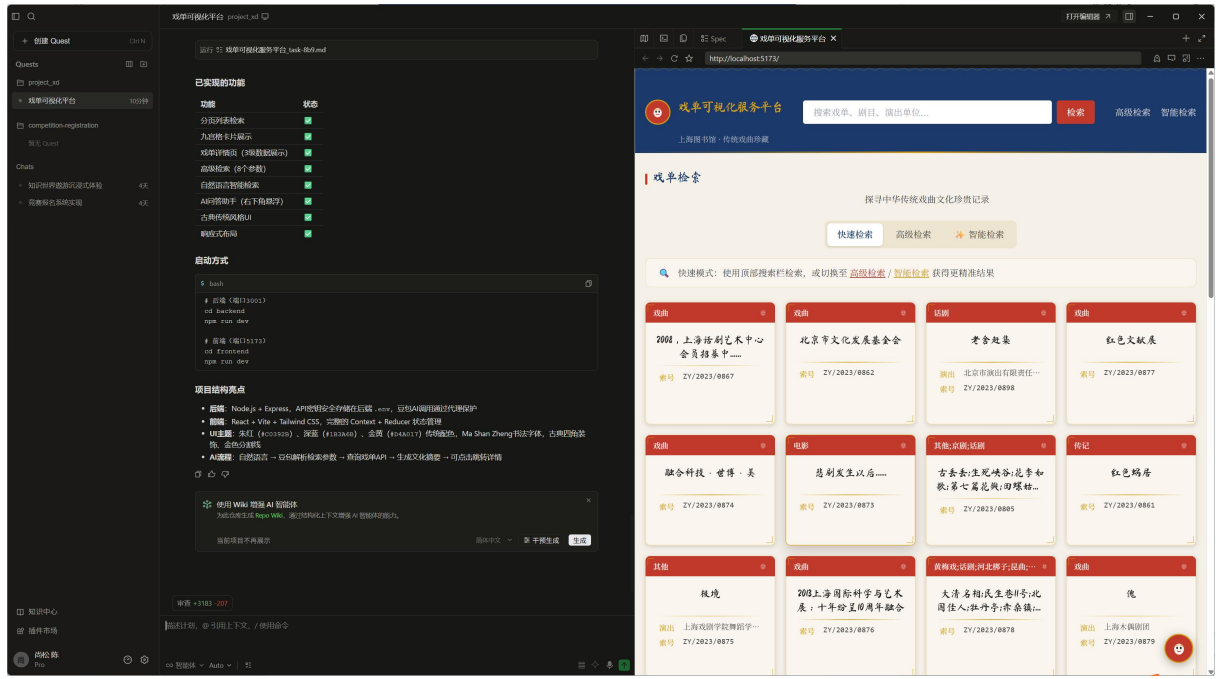


核心流程三：Quest需求澄清



核心流程四：执行开发


戏单可视化平台 project_xd
运行 戏单可视化服务平台_task-8b9.md
SearchContext.jsx +114 A
ChatContext.jsx +57 A
创建前端 Service 层
api.js +40 A
xdApi.js +28 A
aiApi.js +23 A
创建 Hooks
useXdList.js +47 A
useXdDetail.js +50 A
useChat.js +53 A
Task 5: 前端 - Common 公共组件 & Layout 组件
LoadingSpinner.jsx +27 A
ErrorMessage.jsx +27 A
EmptyState.jsx +23 A
Tag.jsx +21 A
现在创建 Layout 组件:
Header.jsx +127 A
Footer.jsx +22 A
PageWrapper.jsx +21 A
Spec
进展
Task 1: 初始化项目脚手架（后端+前端）
Task 2: 后端 - 戏单API代理路由
Task 3: 后端 - 豆包AI代理路由
Task 4: 前端 - 全局主题与状态管理
Task 5: 前端 - Layout组件
Task 6: 前端 - 检索组件
产物
Spec 1
戏单可视化服务平台_task-8b9.md
变更文件 31
.env +7
.gitignore +2
index.js +15
xdService.js +45
xd.js +55
doubaoService.js +162
ai.js +59
引用
暂无引用


戏单可视化平台
已实现的功能
项目结构亮点
戏单检索
戏单可视化服务平台
检索
高级检索
智能检索

SDD — 软件设计描述文档

SDD是AI编码的输入规约，不是写完就束之高阁的文档

SDD要素	戏单项目体现
系统架构	前后端分离：React+Vite / Node.js+Express
组件设计	Layout/Search/List/Detail/AI/Common 六大组件族
接口契约	/api/xd/list、/api/xd/detail/:nid、/api/ai/*
数据模型	戏单三层结构(Event+Instance+Works)、8字段检索参数
非功能约束	CORS白名单、rate-limit 15次/IP/min、防抖300ms
UI/UX规约	传统戏曲风格（朱红/深蓝/金黄）、马善政字体、响应式

关键：SDD的内容会以Prompt/Rule/Memory的形式被Qoder读取并遵守

Harness — 开发脚手架与规约框架

Harness的价值在于先于业务代码存在，定义"代码应该长什么样"

Harness层面	戏单项目实例
项目骨架	frontend/ + backend/ 双目录结构
构建配置	Vite配置、Tailwind配置、PostCSS配置、ESLint配置
运行时脚手架	Express路由注册、中间件加载顺序、axios实例封装
编码规约	组件按功能分目录、Context+useReducer状态模式
安全脚手架	helmet + cors + rate-limit + .env密钥管理
测试脚手架	单元测试 + API集成测试（待建设）

Harness建立的5大约束：

1、路由按资源分离

2、组件按功能域分目录

3、Context+useReducer状态管理

4、API经services层封装

5、.env管理环境变量

Qoder — AI代码编辑器

核心能力

1 对话式编码

用自然语言描述需求，AI生成代码

人在Qoder中输入需求描述，AI自动完成文件创建、代码编写、依赖安装

2 上下文感知

自动读取项目结构、已有代码、Memory规约

AI开发前自动搜索代码库，理解现有架构和编码模式

3 多工具协同

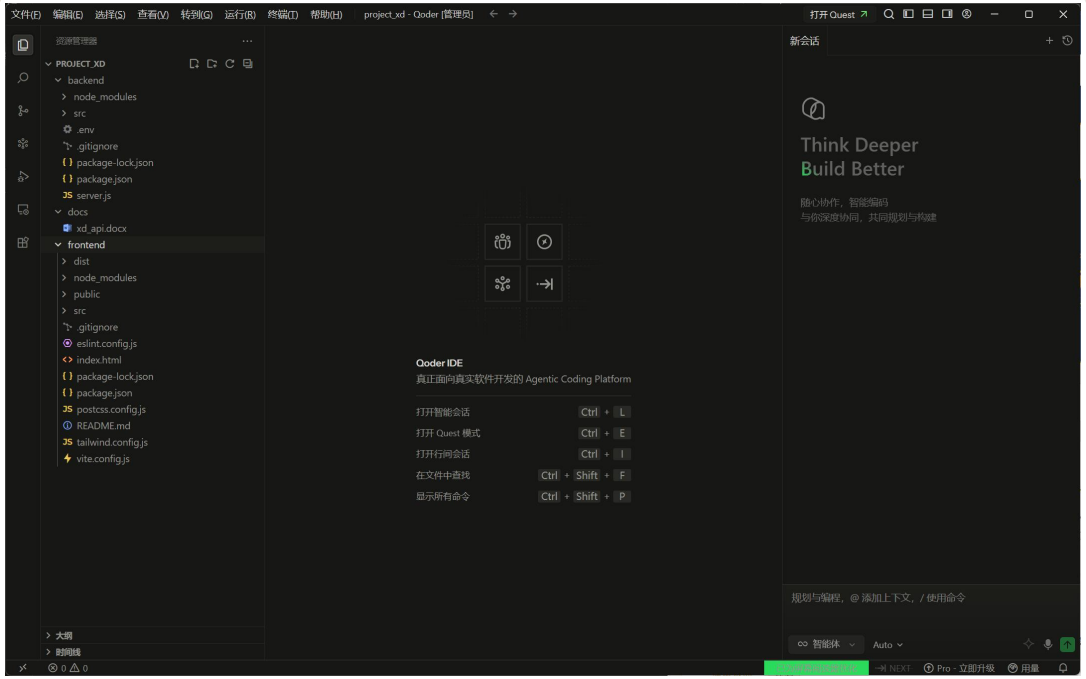
搜索代码库、读写文件、执行终端、LSP智能

SearchCodebase → Edit → Terminal 多工具协作完成复杂任务

4 Memory机制

将SDD要素、项目约定、踩坑经验持久化

如'禁用performance_time参数'写入Memory后AI不再犯



Qoder IDE — 完整项目文件树

Quest — 需求驱动引擎

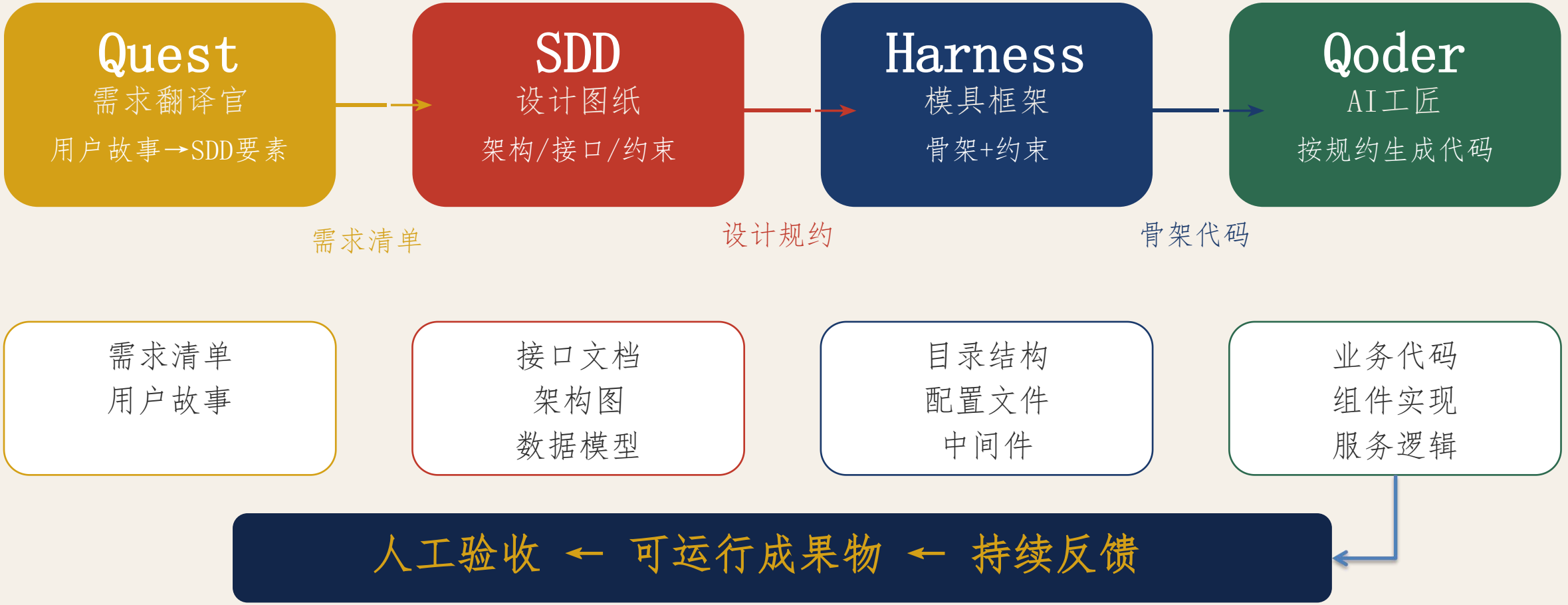
Quest是从需求到设计的"翻译器"

1. 将用户故事/需求描述转化为结构化的SDD要素
2. 辅助梳理功能边界、接口设计、数据流
3. 多轮澄清需求歧义（API Key、技术栈、部署方式）
4. 生成Qoder可消费的开发指令（Prompt/Task）
5. 自动拆解任务：43个文件、10个开发任务

Quest结构化示例：

用户故事	Quest结构化输出	Qoder指令
"自然语言搜戏单"	功能:NL检索; 禁用performance_time	创建POST /api/ai/nl-to-params
"界面要古风"	UI风格:传统戏曲配色	按朱红/深蓝/金黄生成古典组件

四者关系图 — 协同工作模型



03

Vibe Coding戏单项目案例实践过程

十阶段开发流程 — SDD/Harness/Qoder/Quest 参与映射

1. 需求分析
Quest主导

2. 概要设计
SDD主导

3. 接口设计
SDD主导

4. 项目初始化
Harness主导

5. 后端开发
Qoder主导

6. 前端开发
Qoder主导

7. AI集成
Qoder主导

8. UI主题
Qoder主导

9. 联调测试
人工+Qoder

10. 部署交付
Qoder执行

需求描述 → 骨架生成 → 功能迭代 → 审查修正 → 交付

全景映射表 — 四者在各阶段的职责

阶段	SDD角色	Harness角色	Qoder角色	Quest角色	成果物
1. 需求分析	—	—	—	主导	需求清单、用户故事
2. 概要设计	主导	提供架构模板	辅助调研	映射设计要素	架构图、技术选型表
3. 接口设计	主导	约定RESTful规范	辅助生成文档	校验完整性	API文档、数据模型
4. 项目初始	输出技术栈	主导生成骨架	执行创建	—	package.json、目录树
5. 后端开发	输出接口契约	提供分层骨架	主导生成代码	—	routes/services/middleware
6. 前端开发	输出组件树	提供组件模板	主导生成代码	—	pages/components/hooks
7. AI集成	输出Prompt设计	提供AI路由模板	主导生成服务	—	doubaoService/ChatBot
8. UI主题	输出配色方案	提供CSS变量机制	主导生成样式	—	index.css/App.css
9. 联调测试	输出验收标准	提供测试脚手架	辅助调试	—	测试报告、修复记录
10. 部署交付	输出部署架构	提供构建脚本	执行部署命令	—	生产环境产物

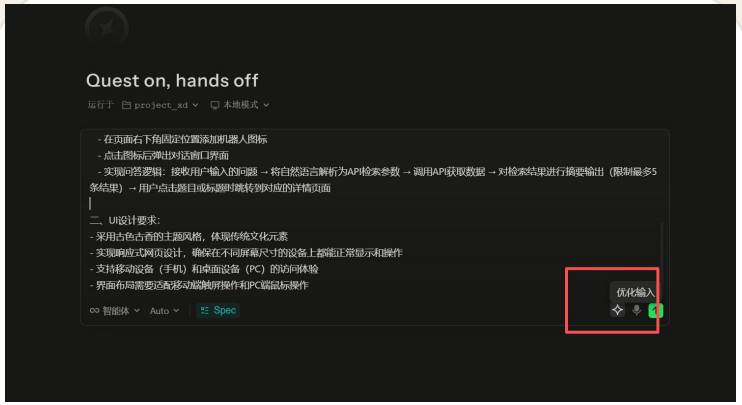
阶段1-3：需求分析 → SDD设计

用户原始需求

"我要做一个戏单可视化系统，能搜京剧越剧，能AI问答，界面要古色古香，数据源结合上海图书馆开放数据API，需要列表、详情、AI问答功能"

Quest转化

- 功能域：检索(快速/高级/NL)、展示(列表/详情)、AI(问答)
- 非功能：传统文化UI、响应式、API Key安全
- 系统边界：基于上海图书馆开放API
- 技术约束：React + Node.js + 豆包大模型



Spec启动界面

Quest多轮需求澄清（API Key、技术栈、部署方式）

戏单相关 API【2026】

返回格式说明

（1）成功返回

```
json
{
  "status": "success",
  "data": {},
  "total": 0
}
```

（2）失败返回

```
json
{
  "status": "error",
  "msg": "错误信息"
}
```

1 接口（戏单列表）

请求方式：GET

接口路径：https://data1.library.sh.cn/webapi/xd/list

请求参数（Query）

参数名	类型	是否必填	默认值	说明
key	String	是		用户 API key
current_page	int	否	1	当前页码
page_size	int	否	20	每页条数，最大 100
keywords	string	否	-	全文检索关键字，模糊匹配
event_name	string	否	-	事件名称，模糊匹配
instance_title	string	否	-	载体题名，模糊匹配
call_number	string	否	-	索取号，模糊匹配
works_name	string	否	-	作品名称，模糊匹配
performance_unit	string	否	-	演出单位，模糊匹配
performance_time	string	否	-	演出时间，模糊匹配
work_type	string	否	-	演出作品类型，模糊匹配

成功返回（data 数组）

阶段1-3：需求分析 → SDD设计

用户原始需求

"我要做一个戏单可视化系统，能搜京剧越剧，能AI问答，界面要古色古香，数据源结合上海图书馆开放数据API，需要列表、详情、AI问答功能"

Quest转化

- 功能域：检索(快速/高级/NL)、展示(列表/详情)、AI(问答)
- 非功能：传统文化UI、响应式、API Key安全
- 系统边界：基于上海图书馆开放API
- 技术约束：React + Node.js + 豆包大模型

戏单相关 API【2026】

返回格式说明

（1）成功返回

```
json
{
  "status": "success",
  "data": {},
  "total": 0
}
```

（2）失败返回

```
json
{
  "status": "error",
  "msg": "错误信息"
}
```

1 接口（戏单列表）

请求方式：GET
接口路径：https://data1.library.sh.cn/webapi/xd/list
请求参数（Query）

参数名	类型	是否必填	默认值	说明
key	String	是		用户 API key
current_page	int	否	1	当前页码
page_size	int	否	20	每页条数，最大 100
keywords	string	否	-	全文检索关键字，模糊匹配
event_name	string	否	-	事件名称，模糊匹配
instance_title	string	否	-	载体题名，模糊匹配
call_number	string	否	-	索取号，模糊匹配
works_name	string	否	-	作品名称，模糊匹配
performance_unit	string	否	-	演出单位，模糊匹配
performance_time	string	否	-	演出时间，模糊匹配
work_type	string	否	-	演出作品类型，模糊匹配

成功返回（data 数组）

2 接口（戏单详情）

基础信息
请求方式：GET
接口路径：https://data1.library.sh.cn/webapi/xd/detail
请求参数（Query）

参数名	类型	是否必填	说明
key	String	是	用户 API key
nid	int	是	戏单列表接口返回结果 nid 属性值

返回结构
data 为数组，包含 3 个顶层分组：**Event（演出事件） → Instance（载体信息） → PerformanceWork（演出作品，数组）**

1. Event 演出事件

一级字段	类型	中文说明	二级字段	中文说明
event_name	字符串	事件名称	—	—
event_session	字符串	届次	—	—
event_year	字符串	发生年	—	—
event_place	字符串	发生地	—	—
event_creator_des	字符	事件编	—	—

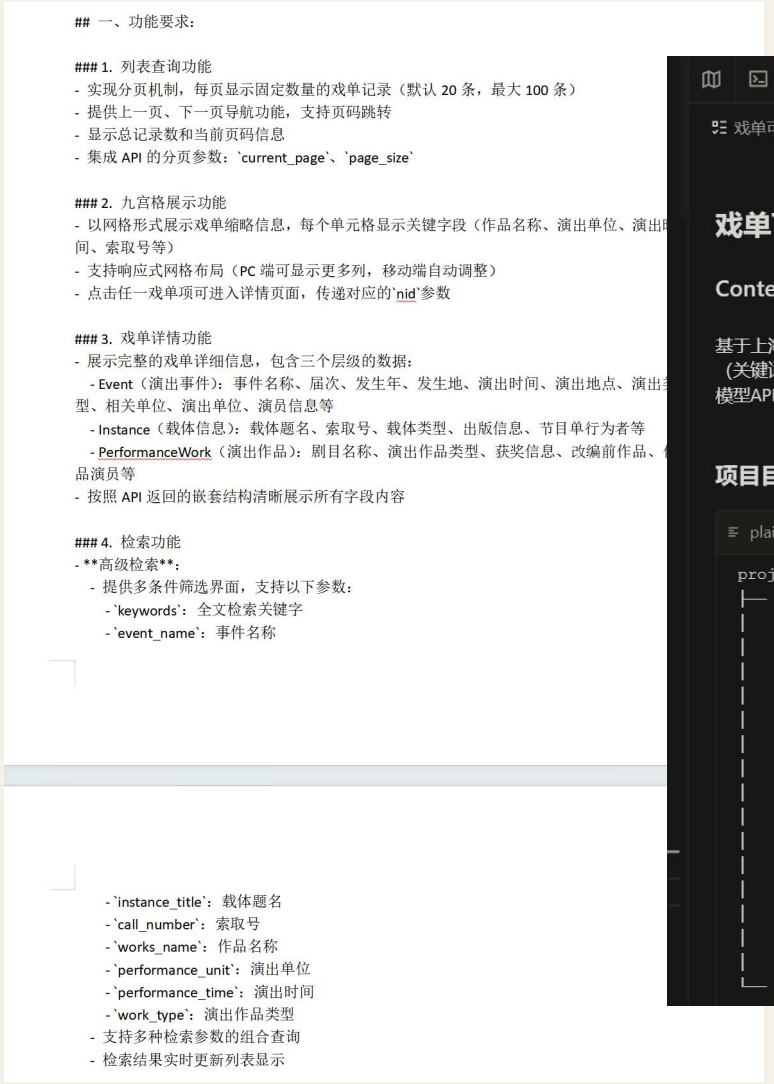
阶段1-3：需求分析 → SDD设计

用户原始需求

"我要做一个戏单可视化系统，能搜京剧越剧，能AI问答，界面要古色古香，数据源结合上海图书馆开放数据API，需要列表、详情、AI问答功能"

Quest转化

- 功能域：检索(快速/高级/NL)、展示(列表/详情)、AI(问答)
- 非功能：传统文化UI、响应式、API Key安全
- 系统边界：基于上海图书馆开放API
- 技术约束：React + Node.js + 豆包大模型



一、功能要求：

1. 列表查询功能
- 实现分页机制，每页显示固定数量的戏单记录（默认 20 条，最大 100 条）
- 提供上一页、下一页导航功能，支持页码跳转
- 显示总记录数和当前页码信息
- 集成 API 的分页参数：`current_page`、`page_size`

2. 九宫格展示功能
- 以网格形式展示戏单缩略信息，每个单元格显示关键字段（作品名称、演出单位、演出时间、索取号等）
- 支持响应式网格布局（PC 端可显示更多列，移动端自动调整）
- 点击任一戏单项可进入详情页面，传递对应的`nid`参数

3. 戏单详情功能
- 展示完整的戏单详细信息，包含三个层级的数据：
  - Event（演出事件）：事件名称、届次、发生年、发生地、演出时间、演出地点、演出类型、相关单位、演出单位、演员信息等
  - Instance（载体信息）：载体题名、索取号、载体类型、出版信息、节目单行为者等
  - PerformanceWork（演出作品）：剧目名称、演出作品类型、获奖信息、改编前作品、作品演员等
- 按照 API 返回的嵌套结构清晰展示所有字段内容

4. 检索功能
- **高级检索**：
  - 提供多条件筛选界面，支持以下参数：
    - `keywords`：全文检索关键字
    - `event_name`：事件名称
    - `instance_title`：载体题名
    - `call_number`：索取号
    - `works_name`：作品名称
    - `performance_unit`：演出单位
    - `performance_time`：演出时间
    - `work_type`：演出作品类型
  - 支持多种检索参数的组合查询
  - 检索结果实时更新列表显示

Spec

戏单可视化服务平台_task-8b9....

戏单可视化服务平台

Context

基于上海图书馆戏单数据开放API，构建一个具有传统戏曲文化美学的可视化检索平台。用户需要能通过多种方式（关键词、高级筛选、自然语言、AI问答）检索戏单数据，并以美观的九宫格卡片和详情页形式呈现内容。豆包大模型API密钥须通过后端代理保护。

项目目录结构

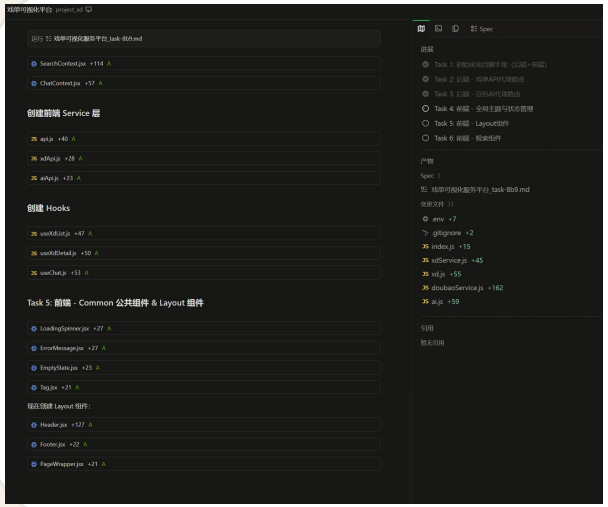
plaintext

```
project_xd/
├── backend/                    # Node.js + Express 后端
│   ├── package.json
│   ├── .env                    # 存储豆包API密钥（不提交git）
│   ├── server.js               # 入口文件
│   └── src/
│       ├── config/index.js     # 环境变量统一读取
│       ├── routes/
│       │   ├── index.js        # 路由聚合
│       │   ├── ai.js           # 豆包API代理路由
│       │   └── xd.js           # 戏单API透传路由（注入key）
│       ├── services/
│       │   ├── doubaoService.js  # 豆包API调用（NL转参、摘要）
│       │   └── xdService.js      # 戏单API调用逻辑
│       └── middleware/
│           └── errorHandler.js   # 统一错误处理
│
└── frontend/                   # React + Vite 前端
```

阶段4：项目初始化 — Harness骨架生成

Qoder自动执行

1. npm create vite@latest frontend -- --template react
2. npm init -y（后端）
3. 安装所有依赖包
4. 生成Tailwind、PostCSS、ESLint配置
5. 创建目录结构和占位文件



代码生成进度（31个文件变更）

Harness建立的约束

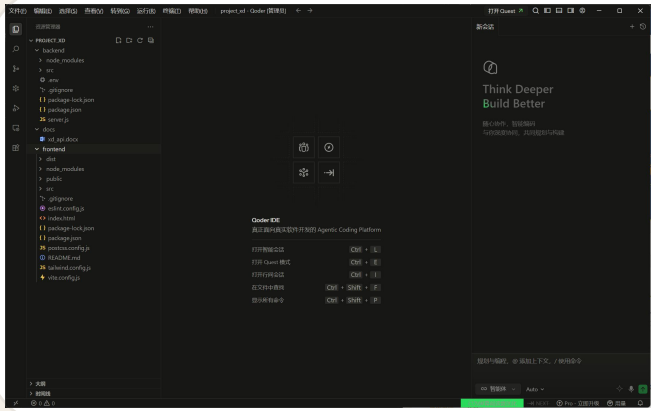
① 后端路由按资源分离（xd.js / ai.js）

② 前端组件按功能域分目录，不允许扁平化

③ 状态管理统一Context+useReducer，不用Redux

④ API调用统一经services/层封装

⑤ 环境变量通过.env管理，不硬编码



最终项目文件树

阶段5：后端核心开发

典型交互示例 — 生成AI代理路由

人在Qoder中的输入：

"按照SDD设计，创建AI代理路由。需要nl-to-params和chat两个接口，chat走三阶段流程：NL解析→戏单检索→摘要生成。限流15次/IP/分钟。"

Qoder自动完成：

1. 读取Memory中的SDD规约（API Key不出后端、豆包OpenAI兼容接口）
2. 读取Harness约束（路由分层、中间件加载顺序）
3. 生成 routes/ai.js（路由定义 + 限流中间件）
4. 生成 services/doubaoService.js（含NL_TO_PARAMS和CHAT_SUMMARY的Prompt）
5. 生成错误处理中间件

人需要审查：

Prompt模板准确性 | 限流参数合理性 | .env配置与测试

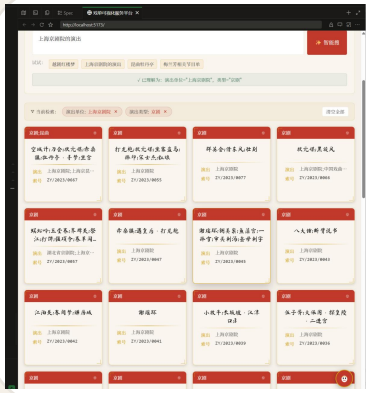
Spec任务详情与验证计划

阶段6：前端核心开发

前端架构与组件

- 页面：ListPage（列表）、DetailPage（详情）
- 组件族：Layout / Search / List / Detail / AI / Common
- 状态管理：React Context + useReducer
- 服务层：api.ts 统一API调用封装
- 路由：React Router 7
- 样式：Tailwind CSS 3 + CSS变量（戏曲主题）

戏单可视化服务平台 — 主页面



智能检索与筛选标签(chips)

详情页 — 事件/载体/作品标签页

2003沪剧详情页

阶段7：AI智能对话集成

三阶段对话流程

阶段1 **NL解析**

自然语言→结构化参数

禁用performance_time

work_type从10个枚举选

→

阶段2 **戏单检索**

调用上海图书馆API

分页处理 page_size≤100

三层数据解析

→

阶段3 **摘要生成**

AI生成自然语言回答

正则提取JSON块

处理非标准返回

AI对话功能要点

✓ 集成豆包大模型（doubao-seed-2-0-pro）

✓ 自然语言检索：解析用户意图为查询参数

✓ 上下文记忆：多轮对话保持连贯

✓ 快捷示例：预设常见查询场景

AI智能对话 — 自然语言检索

阶段8：戏曲主题UI实现

传统戏曲风格配色方案

朱红 #C0392B	深蓝 #1B3A6B	金黄 #D4A017
米白 #F5F0E8	墨黑 #1C1C1C	玉绿 #2D6A4F

Harness提供：Tailwind + CSS变量机制

Qoder执行：按主题规范生成 index.css、App.css、组件内联样式

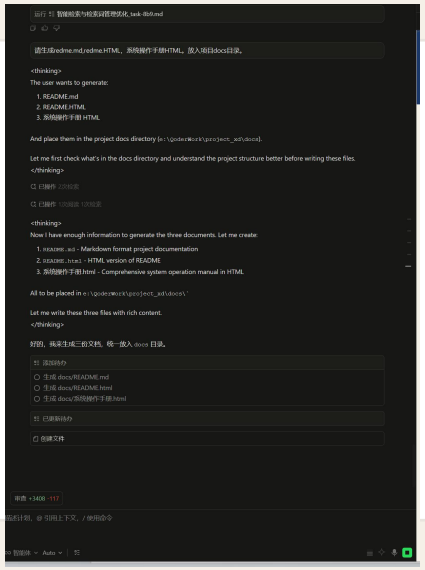
戏曲主题主页面效果

优化迭代过程

阶段9-10：联调测试与部署交付

联调测试

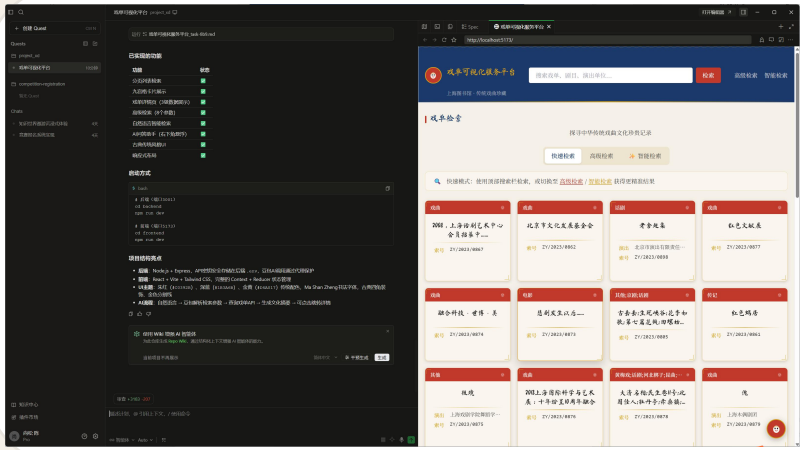
1. 启动后端 npm run dev → 验证 /api/xd/health 返回OK
2. 启动前端 npm run dev → 验证页面能打开
3. 智能检索输入"越剧红楼梦" → 确认AI解析正确
4. 快捷示例按钮逐一确认返回非空结果
5. 检索后确认页面显示检索词标签(chips)
6. 点击chip的×确认条件被删除、列表自动刷新
7. AI助手输入"1990年代越剧演出"确认不再用performance_time



Qoder任务管理 — 文档生成

交付清单

- ✓ 完整的前后端源代码
- ✓ README.md / README.html 项目文档
- ✓ 系统操作手册.html
- ✓ API接口文档



开发完成 — 分屏展示

04

Vibe Coding实践复盘

实战复盘：四者协作的四个阶段

阶段一

需求→SDD

Quest主导

输入：用户口头描述

Quest转化：提取功能域、非功能需求

产出：结构化需求清单、SDD要素

阶段二

SDD→Harness

人工+Qoder辅助

SDD输出：目录结构设计

Qoder执行：创建项目、安装依赖

Harness建立：5大约束规则

阶段三

SDD+Harness→代码

Qoder主导

读取Memory中的SDD规约

读取Harness约束

生成路由/服务/中间件/组件代码

阶段四

验收→Memory沉淀

人工+Qoder

发现问题→沉淀为Memory

API Key不出后端等

避坑指南 — SDD相关

SDD常见陷阱与避坑方法

坑1 SDD过于笼统

表现：AI生成的代码偏离需求

避坑：SDD中每个功能点必须包含：输入、输出、校验规则、异常处理

坑2 SDD不包含约束

表现：AI使用了禁用字段（如performance_time）

避坑：在SDD中显式标注"禁止项"，并在Qoder Memory中写入

坑3 SDD与实际API不一致

表现：前端调后端接口参数对不上

避坑：SDD的接口设计必须以API文档为准，先验证再写SDD

坑4 SDD中遗漏非功能需求

表现：忘记限流、忘记CORS、忘记密钥保护

避坑：使用SDD检查清单：安全/性能/可用性/可维护性

避坑指南 — Harness & Qoder相关

Harness相关避坑

约束不足	AI生成代码结构混乱	先建Harness（目录模板+空文件），再让AI填充
与AI上下文脱节	AI不知道有Context模式	将Harness规约写入Qoder Memory
过度Harness	脚手架太重，AI代码塞不进去	Harness只定义骨架和约束，不预设实现细节

Qoder使用避坑

一次生成整个项目	代码量大难审查	分模块、分文件生成，每步验证后再继续
不写Memory	AI每次都从零开始	每发现一个坑立即写入Memory
Prompt不够具体	AI生成通用代码	Prompt中引用SDD具体条款
不审查直接用	潜在bug上线	AI生成代码后必须人工走读关键逻辑
忽略AI搜索建议	找不到正确上下文	让AI先SearchCodebase再生成

戏单项目特有避坑

performance_time字段为空
查询返回0条 → 年份词放入keywords

work_type非标准值
只有10个有效枚举 → Prompt中硬编码

豆包API Key泄露
前端直接调用 → 必须经后端代理

API分页不一致
page_size>100被截断 → 前端限制最大100

AI返回非JSON
豆包偶尔返回带说明文字 → 正则提取{...

CORS跨域
前后端端口不同 → 后端cors白名单配置

高效协作最佳实践

1 SDD先行　先完成SDD → 再让Qoder按SDD逐模块生成

没有SDD约束，AI会按通用方案生成，不符合项目特定需求

2 Harness骨架优先　先用Harness建好骨架 → 再让AI在骨架内填充

骨架存在后，AI生成的代码会自然遵循已有的分层模式

3 逐步生成+即时验证　按模块分步生成，每步验证运行后再继续

后端骨架→后端API→前端骨架→列表页→详情页→AI路由→聊天→UI主题

4 Memory持续沉淀　每个坑都写入Qoder Memory，确保AI后续不再犯，**是团队知识资产，避免重复踩坑，加速团队成长**

如'禁用performance_time'、'AI必须通过后端代理'等

5 Quest需求结构化　用户故事→Quest结构化→SDD条目→Qoder编码

避免用户故事直接丢给Qoder编码导致需求遗漏

Vibe Coding 实践核心流程

完整的SDD × 合理的Harness × 精准的Qoder Prompt × 持续的Memory沉淀

= 高质量交付

SDD是图纸，Harness是模具，Qoder是工匠，Quest是翻译官

没有图纸和模具，工匠只能凭感觉干活；

有了图纸和模具，工匠才能高效精准地产出合格产品。

Vibe Coding开发行动规范

- 每个项目启动前，先完成SDD（至少覆盖架构、接口、约束）
- SDD完成后，先建Harness骨架，再让AI填充
- 每个坑都写入Qoder Memory，不要只在对话中修复
- AI生成代码必须人工审查，特别是Prompt模板和安全逻辑
- 分模块逐步生成，每步验证后再继续下一步
- 定期回顾Memory，清理过时条目，补充新发现

05

进阶篇：引入 Loop Engineering

进阶篇

引入Loop后的戏单项目开发流程

前面讲的是"人驱动AI"模式：SDD + Harness + Qoder 协作开发

关键升级：让Qoder先根据SDD生成Judge，再生成代码，Loop自主验收迭代

传统 SDD+Harness+Qoder 流程

① 人工编写SDD → 人工审核

② 人工建Harness → 人工审核

③ 人工写Prompt → Qoder生成 → 人工验收

④ 发现问题 → 人工修改Prompt → 重新生成

⑤ 每个模块重复③④，全程人工驱动

引入 Loop 后的流程

① 人工编写SDD + Harness（规则设计）

② Qoder根据SDD自动生成Judge脚本

③ Qoder根据SDD+Harness生成业务代码

④ Loop自验：Judge跑代码 → 不通过则自主修复

⑤ 人只需：设计规则 + 审查最终结果

核心转变：从"人一步步驱动AI" → "人设计规则，AI自主运转"

核心流程与演进路径

SDD × Harness × Qoder × Memory × Loop

= 高质量AI工程交付

AI工程四阶段演进

Prompt Engineering → Context Engineering → Harness Engineering → Loop Engineering

SDD是图纸，Harness是模具，Qoder是工匠，Memory是经验，Loop是自动化工厂

没有图纸和模具，工匠只能凭感觉干活；

有了图纸和模具，工匠能高效精准地产出合格产品；

而Loop工程，让工厂24小时自主运转，人只需设计规则、审查结果。

从"人驱动AI"变为"AI自主运转"

Judge代码样例 — 戏单项目三类Judge具体实现

Judge就是普通测试脚本，放在项目 tests/ 目录下，用Python/JS均可

结构Judge

结构Judge — tests/

```
FORBIDDEN = ["performance_time"]
ALLOWED_DIRS =
["routes/","services/","components/"]
def check_structure():
    for f in
Path("src").rglob("*.js"):
        assert any(d in str(f) for d
in ALLOWED_DIRS)
        code = f.read_text()
        for field in FORBIDDEN:
            assert field not in code,
```

检查目录结构 + 禁用字段（Harness约束）

功能Judge

功能Judge — tests/

```
def test_xd_list_api():
    r = requests.get(

"http://localhost:3001/api/xd/list",
    data = r.json()
    # 检查SDD契约
    assert "events" in data
    assert all("nid" in e for e in
data["events"])
    assert r.status_code == 200
```

检查API返回格式（SDD接口契约）

体验Judge

体验Judge — tests/

```
def test_theme_compliance():
    css = Path("src/styles/theme.css")
        .read_text()
    # 检查戏曲主题配色
    assert "#C0392B" in css  # 朱红
    assert "#1B3A6B" in css  # 深蓝
    assert "#D4A017" in css  # 金黄
    # 检查仿宋字体
    assert "FangSong" in css
```

检查主题配色+字体（设计规范）

Judge = 把SDD/Harness中的文字约束转化为可执行的assert断言，Loop用它们自动验收

Loop实战 — Judge优先生成的自主闭环示例

以"戏单列表API"为例：Qoder先根据SDD生成Judge，再生成代码，Loop自验

Plan

读取SDD：/api/xd/list规约

读取Harness：路由在routes/xd.js

读取Memory：禁用performance_time

→

Do-Judge

Qoder根据SDD先生成Judge

生成API契约断言脚本

生成结构检查脚本

Judge就绪，等待代码

→

Do-Code

Qoder生成xd.js路由代码

生成xdService.js服务层

参数校验、分页、错误处理

调用上海图书馆API

→

Check+Act

Judge自动验收代码

结构/功能/安全全检查

不通过→自主修复→重验

全部通过→输出给人审查

Judge在代码之前生成（类似TDD），SDD既是代码规约也是Judge规约

SDD层	接口契约、字段约束、禁止项 → 同时生成代码+Judge	
Harness层	目录结构、路由规约、中间件顺序 → 结构Judge自动检查	
Memory层	踩坑经验、API特性、修复模式 → Loop自主迭代时参考	

从戏单到大型项目 — Loop工程规模化

分层Judge体系

小项目：1-2个Judge即可

大型项目：单元Judge + 集成Judge

+ 系统Judge + 安全Judge

戏单：3类Judge → 大型：分层级联

SDD分解策略

小项目：单份SDD覆盖全局

大型项目：SDD按子系统分解

每个子系统独立Loop运行

戏单：1份SDD → 大型：N份SDD

并行Loop编排

小项目：单Loop串行开发

大型项目：多Loop并行+依赖编排

前端Loop / 后端Loop / 测试Loop

戏单：1个Loop → 大型：多Loop协同

人工介入节点

小项目：最终人工审查

大型项目：关键节点人工审批

架构变更/安全策略/接口契约

戏单：末尾审查 → 大型：关卡审批

Loop工程实施路线图

第一步
戏单小型项目
手动验收→写Judge

→

第二步
中型项目实践
单Loop+3类Judge

→

第三步
大型项目推广
多Loop并行+分层Judge

→

全自动形态
全自主Loop工厂
人只设计规则和审批

谢谢

陈尚松 | 上海福呈数据科技有限公司