

Zero-DevToolbox 技术文档

通用、低耦合、跨项目复用的 AI 辅助开发 Skill 参考

项目或主题: Zero-DevToolbox

作者: 项目组

日期: 2026 年 8 月 15 日

版本: v1.0

本文档说明 Zero-DevToolbox 的项目定位、Skill 架构、主要 workflows、安装方式、知识库机制、配套配置与已知限制。

目录

1 文档概述	3
2 项目简介	3
2.1 设计目标	3
3 技术栈与运行环境	4
4 目录结构	4
5 总体架构	5
5.1 分层组成	5
5.2 典型执行流程	5
6 Skill 模块参考	6
6.1 代码维护类工作流	6
6.2 理解与文档类工作流	7
6.3 Git 交付工作流	7
7 项目知识库与上下文连续性	7
7.1 初始化	7
7.2 日常更新	7
7.3 解决的问题	7
8 LaTeX 文档生成模块	7
8.1 工作模式	7
8.2 模板与输出	8
9 安装与使用	8
9.1 克隆仓库	8
9.2 复制全部 Skill	8
9.3 调用示例	8
10 配套配置	9
10.1 Cursor 行为规则	9

10.2 Ignore 模板	9
10.3 EditorConfig	9
11 构建、运行与测试	9
12 常见问题	9
13 已知限制	10
14 许可证与来源	11

1 文档概述

本文档面向 Zero-DevToolbox 的使用者与维护者，基于当前工作树中的 README、Skill 定义、元数据、模板、Cursor 规则、ignore 模板、EditorConfig、许可证及 Git 信息整理。分析范围采用默认的 `scope=core`，重点描述对外可复用的 workflow、配置结构和使用路径。

范围说明

Zero-DevToolbox 是配置与文档资源仓库，不包含传统应用程序入口、业务类、运行时服务或公共 API。因此本文不虚构类图、函数签名、数据库结构、性能指标或部署流程，而是将 Skill 目录视为主要功能模块。

2 项目简介

Zero-DevToolbox 是一套简洁、通用、低耦合、可跨项目复用的 AI 辅助开发 Skill 集合。与绑定单一项目架构、框架或业务规则的 Skill 不同，本仓库提供可迁移的工作流定义；Skill 在执行时读取目标项目的代码、配置、测试、AGENTS.md 和文档，再确定具体操作。

当前仓库包含 9 个需要显式调用的 workflow Skill，主要解决以下开发问题：

- 代码注释不足、局部实现复杂以及重复代码逐渐累积；
- 功能调用链、数据流、状态变化、单位和坐标系难以快速理解；
- README 与真实代码或配置长期不同步；
- Git 提交、提交信息生成和推送过程重复；
- Codex 跨会话后容易丢失项目背景、决策、问题和进度；
- 零散笔记或代码库知识缺少结构化技术文档。

2.1 设计目标

- **低耦合**：不写死目标项目的模块、框架、数据模型或业务逻辑。
- **可迁移**：可以复制整套 `.agents/skills/`，也可以只复制单个 Skill 目录。
- **职责单一**：每个 Skill 聚焦一个明确 workflow，并通过显式调用启动。
- **安全可验证**：Skill 定义中包含范围控制、已有改动保护和结果验证规则。
- **上下文连续**：通过结构化、可版本管理的项目知识库保存长期有效信息。

3 技术栈与运行环境

组成	用途
Markdown	保存 Skill 主指令、README 和工作流参考文档。
YAML	<code>agents/openai.yaml</code> 保存显示名称、默认提示词和调用策略。
LaTeX	提供可直接上传 Overleaf 的中文技术文档模板。
Cursor MDC	<code>.cursor/rules/karpathy-guidelines.mdc</code> 保存始终应用的编码行为规则。
EditorConfig	统一字符编码、换行、缩进和行尾空格行为。
Git	获取仓库状态，也是 <code>\$commit</code> 工作流的基础。

仓库自身没有包管理器、运行时依赖清单、编译脚本或自动测试入口。使用 Skill 需要一个能够从 `.agents/skills/` 发现项目级 Skill 的 AI 编码工具。生成的 LaTeX 文档默认面向 Overleaf 的 XeLaTeX 编译器。

4 目录结构

```
Zero-DevToolbox/
|-- .agents/
|   |-- skills/
|       |-- add-code-comments/
|       |-- commit/
|       |-- generate-latex-document/
|       |   |-- agents/openai.yaml
|       |   |-- assets/document-template.tex
|       |   |-- references/codebase-workflow.md
|       |   |-- references/notes-workflow.md
|       |   |-- SKILL.md
|       |-- init-project-knowledge/
|       |-- optimize-target-code/
|       |-- readme/
|       |-- reduce-code-redundancy/
|       |-- trace-code-flow/
|       |-- update-project-knowledge/
|-- .cursor/rules/karpathy-guidelines.mdc
|-- ignore/
|   |-- .gitignore
|   |-- .claudeignore
```

```
|  `-- .cursorignore  
|-- .editorconfig  
|-- LICENSE  
|-- README.md  
`-- README_zh.md
```

除 `generate-latex-document` 外，当前各 Skill 的核心结构是 `SKILL.md` 与 `agents/openai.yaml`。`generate-latex-document` 额外包含模板以及笔记整理、代码库说明两套 workflow 参考。

5 总体架构

5.1 分层组成

1. **发现与交互层：**`agents/openai.yaml` 提供显示名称、简短说明和默认提示词；所有现有 Skill 都设置 `allow_implicit_invocation: false`。
2. **workflow 定义层：**`SKILL.md` 描述适用场景、执行步骤、范围边界、安全规则和完成报告。
3. **可选资源层：**复杂 Skill 可以通过 `references/` 拆分专项 workflow，通过 `assets/` 保存可复用模板。
4. **目标项目层：**Skill 被复制到目标仓库后，根据该项目的真实代码、配置、测试、Git 状态和文档执行。
5. **配套规则层：**Cursor 规则、ignore 模板和 EditorConfig 为编码行为、上下文过滤和格式统一提供补充支持。

5.2 典型执行流程

1. 将完整 Skill 集合或选定目录复制到目标项目的 `.agents/skills/`。
2. 在 AI 编码会话中使用 `$skill-name` 显式调用所需 workflow。
3. Skill 读取目标项目适用的 `AGENTS.md`、相关代码、配置、测试和文档。
4. Skill 确定处理范围，并避开第三方依赖、生成文件、缓存、日志和无关模块。
5. 根据用户要求执行只读分析或小范围修改。
6. 运行与风险相称的检查，并报告已验证结果、未验证内容及原有工作区改动。

显式调用

现有 9 个 Skill 均禁用隐式调用。自然语言任务不会自动授权这些工作流；使用者需要明确写出对应的 `$skill-name`。

6 Skill 模块参考

Skill	核心职责
<code>\$add-code-comments</code>	为用户指定的代码补充简洁、准确的变量和函数注释，并整理局部多余空行；只允许修改注释和空白格式，主要适用于支持 <code>//</code> 注释的语言。
<code>\$commit</code>	检查 Git 工作区、暂存区、分支和远端，生成符合仓库风格的提交信息，并完成暂存、提交和推送。
<code>\$generate-latex-document</code>	使用内置模板，将零散资料整理为技术文档，或分析代码库生成结构化说明；只输出独立的 <code>.tex</code> 文件。
<code>\$init-project-knowledge</code>	首次分析现有项目，创建或完善 <code>AGENTS.md</code> 和结构化项目知识库，并使用真实代码与资料进行初步填充。
<code>\$optimize-target-code</code>	优化指定函数、类、文件或小范围功能，关注效率、复杂度、数值稳定性、健壮性和可读性，并保持外部行为。
<code>\$readme</code>	根据真实代码、配置、脚本和文档，同步创建、检查或更新英文 <code>README.md</code> 与中文 <code>README_zh.md</code> 。
<code>\$reduce-code-redundancy</code>	审计重复实现、未使用代码、冗余文件和分散工具；只清理证据充分且能够验证的候选项。
<code>\$trace-code-flow</code>	只读追踪指定功能从入口到输出的调用链、数据流、状态变化、外部交互、单位和坐标系。
<code>\$update-project-knowledge</code>	检查已完成工作，仅在产生经过确认且长期有效的信息时更新已有项目知识库。

6.1 代码维护类工作流

`$add-code-comments`、`$optimize-target-code` 和 `$reduce-code-redundancy` 共同覆盖可读性、局部实现质量与仓库级冗余治理。它们都强调限定范围、保留外部行为、避免顺带重构，并保护执行前已经存在的用户改动。

冗余清理支持 `mode=audit`（只分析）、`mode=apply`（清理已确认候选项）以及默认的“先审计、后确认”模式。

6.2 理解与文档类工作流

`$trace-code-flow` 用于建立从入口到输出的代码理解，重点区分接口、实现、调用者、动态路径、状态、副作用和数值约定。`$readme` 将项目事实同步到中英文入口文档。`$generate-latex-document` 则将笔记或代码库整理成适合交付的技术文档。

6.3 Git 交付工作流

`$commit` 会检查未暂存与已暂存差异、近期提交风格、当前分支和远端信息。其目标是根据实际 diff 生成提交信息，并安全完成提交和推送，而不是盲目提交整个工作区。

7 项目知识库与上下文连续性

7.1 初始化

`$init-project-knowledge` 面向首次建立知识库的场景。它会检查现有 `AGENTS.md`、文档索引和已有知识文档，优先复用现有结构；缺少明确约定时，默认使用项目根目录下的 `docs/`。

知识库可以包含项目概览、架构、接口、技术决策、问题记录和当前进度。其目标是形成轻量、结构化、可版本管理、适合 Codex 长期使用的项目上下文。

7.2 日常更新

`$update-project-knowledge` 只更新已有知识库，不擅自创建新结构。它根据当前任务、Git 差异和验证结果，记录以后仍然有用的确认信息；普通聊天、临时想法、未验证猜测和冗长日志不会写入。

7.3 解决的问题

这两个 Skill 形成“首次建立、按需维护”的生命周期，减少跨会话时项目目标、架构、接口、决策、已知问题和进度信息的丢失。这里的“知识库”是保存在项目文档中的版本化知识集合，不是数据库服务，也不引入额外运行时依赖。

8 LaTeX 文档生成模块

8.1 工作模式

- **笔记整理模式：**处理 Markdown、TXT、会议记录、学习笔记及其他零散资料。

- **代码库说明模式：**分析项目入口、核心模块、公共接口、配置、测试和主要流程。

两种模式分别读取 `references/notes-workflow.md` 和 `references/codebase-workflow.md`。除非用户明确要求混合生成，否则一次只使用一个工作流。

8.2 模板与输出

`assets/document-template.tex` 提供 A4 页面、中文排版、页眉页脚、目录、代码块、表格和信息框样式。生成结果需要替换标题、副标题、项目名、作者、版本、封面说明和正文占位符，输出为不依赖 Skill 目录的完整 `.tex` 文件。

工作流不调用本地 LaTeX、XeLaTeX 或 `latexmk`，也不生成 PDF。结果默认上传 Overleaf 后使用 XeLaTeX 编译。

9 安装与使用

9.1 克隆仓库

```
git clone https://github.com/DreamCasterZero/Zero-DevToolbox.git
```

9.2 复制全部 Skill

macOS 或 Linux:

```
mkdir -p /path/to/your-project/.agents  
cp -R Zero-DevToolbox/.agents/skills /path/to/your-project/.agents/
```

PowerShell:

```
New-Item -ItemType Directory -Force C:\path\to\your-project\.agents  
Copy-Item -Recurse Zero-DevToolbox\.agents\skills C:\path\to\your-project\.agents\
```

9.3 调用示例

```
$trace-code-flow trace the checkout flow from entry to output.  
$reduce-code-redundancy mode=audit  
$init-project-knowledge analyze this repository and build its knowledge base.  
$readme synchronize the English and Chinese READMEs.
```

10 配套配置

10.1 Cursor 行为规则

`.cursor/rules/karpathy-guidelines.mdc` 设置为 `alwaysApply: true`, 要求编码前明确假设与权衡、优先使用最简单方案、只进行必要修改, 以及用可验证目标驱动执行。

10.2 Ignore 模板

`ignore/` 提供 `.gitignore`、`.claudeignore` 和 `.cursorignore`。规则覆盖密钥、依赖目录、构建和测试产物、日志、临时文件、数据库、媒体、IDE 元数据及工具缓存。

复制前检查

默认 `ignore` 模板会排除锁文件和图片等内容, 而部分项目应提交这些文件。复制后需要根据目标项目的版本控制策略检查并调整。

10.3 EditorConfig

`.editorconfig` 使用 UTF-8 和 LF, 要求文件末尾换行, 并为 Python、Web 与配置文件、Markdown、Makefile 和 Shell 脚本定义缩进及行尾空格策略。

11 构建、运行与测试

Zero-DevToolbox 不包含可执行应用、包管理配置或自动测试脚本, 因此不存在仓库级依赖安装、构建、启动和测试命令。文档中的复制命令来自当前 README; 本次生成没有实际复制到其他项目, 也没有执行网络克隆。

生成的本文档应上传 Overleaf, 并选择 XeLaTeX 编译器。按照 Skill 约束, 本次没有检查本地 LaTeX 环境, 也没有执行编译或生成 PDF。

12 常见问题

问题: 可以只复制一个 Skill 吗?

可以。每个 Skill 位于独立目录；复制时应保留其中的 SKILL.md、agents/openai.yaml 以及该 Skill 自带的 assets/ 或 references/。

问题：为什么 Skill 不会自动触发？

当前所有 agents/openai.yaml 都设置 allow_implicit_invocation: false, 使分析、修改、提交和知识库维护等工作流只能在用户明确授权后执行。

问题：项目知识库是不是数据库？

不是。它是由 AGENTS.md、文档索引及结构化 Markdown 文档组成的版本化知识集合，用于保存长期有效的项目上下文。

问题：为什么没有类与函数参考？

当前仓库以 Markdown、YAML、MDC 和 LaTeX 资源为主，不包含传统业务源码、类或公共函数，因此本文按 Skill 模块和工作流进行说明。

13 已知限制

- 实际可用性取决于宿主 AI 编码工具是否支持从 .agents/skills/ 发现项目级 Skill。
- \$add-code-comments 主要面向支持 // 注释的语言。
- \$generate-latex-document 只生成 .tex, 不负责本地编译、PDF 生成或 Overleaf 上传。
- 仓库没有自动化测试来验证 Skill 在不同宿主工具、操作系统和目标项目中的行为。
- 当前工作树包含尚未提交的配置迁移和 README 修改；本文描述当前文件状态，而不是最近一次提交的完整快照。

14 许可证与来源

项目采用 MIT License，版权信息为 Copyright (c) 2026 DreamCasterZero。远端仓库地址为：

<https://github.com/DreamCasterZero/Zero-DevToolbox>

本文内容依据当前仓库文件生成，未使用外部资料，也未推断未获支持的版本、性能或兼容性结论。