Claude Skills

2025年10月18日 3 分钟阅读

#Claude #Claude Skills #Context Engineering

> *Claude Skills 是一种模块化的、可重复使用的能力包。你可以把它看作一个文件夹，里面封装了针对特定任务的指令、知识、甚至是可执行的代码脚本。它的核心价值在于，能将通用的 Claude 模型，转变为能够精准执行特定领域任务的"专家"。*

像function calling一样来动态调用不同职业技能

TLDR

Claude Skill本质和Funciton Calling类似，它可以被视为**选择性提示词注入**（selective prompt injection），在对话开始，只是告诉LLM关于skills的元数据（类似Function Call的函数声明），而在实际交互过程中，由 LLM 通过类似工具调用来启动，从而在需要时动态地加载上下文。

引言：告别重复，让AI更懂你

你是否曾感觉，每次与AI对话都像是在教一位全新的实习生？一遍又一遍地解释相同的格式要求、重复提供同样的背景知识，这种低效的沟通循环，正是许多人与AI协作时的痛点。

想象一下，如果你能为AI助手配备一套"专属工具箱"或者一叠"智能秘籍卡片"。每张卡片都封装了一项特定任务的完整指南，无论是撰写符合公司品牌规范的报告，还是分析特定格式的数据。你只需设定一次，AI就能在需要时自动翻阅并精准调用。这，就是 Claude Skills 带来的变革——它将一次性的"提问-回答"模式，升级为持久的"一次设定，自动调用"的智能协作。 另外，有趣的是，基本在同一时间，Manus在它的Context Engineering博客中也公布了类似的技术，只是这个技术用在了Context的管理上，而不是某个角色的技能加载上，但是却给了我一个非常大的提示，这两个技术可以互相组合，从而实现更好的Agentic Context Engineering, 但这是后话，按下不表。

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#Claude #Claude Skills #Context Engineering

本文档的目标，就是为初学者揭开 Claude Skills 的神秘面纱，清晰解释它是什么、如何工作，以及为什么它能彻底改变你与AI协作的方式。

接下来，让我们深入了解 Claude Skills 的真正含义，以及它能为你带来哪些核心优势。

1. 到底什么是 Claude Skills?

Claude Skills 是一种模块化的、可重复使用的能力包。你可以把它看作一个文件夹，里面封装了针对特定任务的指令、知识、甚至是可执行的代码脚本。它的核心价值在于，能将通用的 Claude 模型，转变为能够精准执行特定领域任务的“专家”。

使用 Skills 能带来三大核心优势：

> 专业化 (Specialize Claude): 将 Claude 从一个“什么都懂一点”的通才，训练成特定任务的“专才”。无论是遵循你公司的品牌指南，还是按照特定框架分析数据，Skills 都能让 Claude 的输出精准满足你的要求。
>
> 效率化 (Reduce repetition): 告别在每次对话中重复解释背景和要求的烦恼。通过“一次创建，随处使用”的模式，Skills 能让你把宝贵的时间花在更有创造性的工作上。
>
> 组合化 (Compose capabilities): 多个简单的 Skills 可以像乐高积木一样组合起来，自动协同解决更复杂的工作流程。例如，你可以组合一个“财务数据分析 Skill”和一个“PPT 简报生成 Skill”，一站式完成从数据处理到报告输出的全过程。

为了更清晰地理解其独特价值，我们可以将 Claude Skills 与我们最熟悉的普通提示词（Prompt）进行对比：

特性 普通提示词 (Prompt) Claude Skills 用途 主要用于处理一次性的、对话内的任务。 专为可复用的工作流和专业任务而设计。 调用方式 用户在每次对话中手动输入或粘贴。 AI 根据用户请求的意图自动识别并调用。 持久性 仅在当前单次对话中有效，关闭后即失效。 跨对话持续可用，一次设定，长久生效。

理解了 Skills 的核心价值后，你可能会好奇，Claude 是如何“智能”地在需要时才调用这些能力的呢？这背后是一套精巧的工作机制。

2. Skills 的工作原理：三步智能加载魔法

Claude Skills 的核心工作机制非常巧妙，被称为**“渐进式披露(Progressive Disclosure)”**。

这个概念听起来很专业，但可以用一个简单的比喻来理解：就像一位聪明的图书管理员，Claude 不会一次性把图书馆里所有的书都搬到你面前。相反，它会先查看所有书的“索引卡片”，了解每本书的大致内容。只有当你提出具体需求时，它才会根据索引卡找到对应的书架，并取出那本书的详细内容给你。

这种分层加载的方式，极大地节省了AI的“思考空间”（即上下文窗口），确保了运行的高效性。具体来说，这个过程分为三个层级：

第一层：元数据（始终加载的“索引卡”）

内容： 每个 Skill 的 SKILL.md 文件都包含一个名为 name（名称）和 description（描述）的元数据部分。这部分内容极其简短，只告诉 Claude “我是谁”以及“我能做什么”。

加载时机： 始终加载。在每次对话开始时，Claude 都会加载所有可用 Skills 的元数据，就像图书管理员随身携带的索引卡一样。

空间占用： 极小。每个 Skill 的元数据仅消耗约 ~100 个 Token，这意味着你可以拥有大量 Skills 而不必担心会挤占宝贵的上下文空间。

第二层：指令（按需触发的“说明书”）

内容： SKILL.md 文件的主体部分，包含了完成任务所需的详细步骤、工作流程和最佳实践。

加载时机： 按需触发。只有当用户的请求与第一层元数据中的“描述”相匹配时，Claude 才会真正去读取这份详细的“说明书”。

空间占用： 中等。这部分内容被读取后才会进入上下文窗口（通常小于 5k Tokens）。

第三层：资源与代码（即用即取的“工具箱”）

内容： Skills 文件夹中可以包含额外的辅助文件，如参考文档、数据模板（.md, .json），甚至是可执行的 Python 脚本（.py）。

加载时机： 即用即取。这些资源只有在第二层的指令中被明确引用或调用时，才会被加载或执行。

空间占用： 极其高效。特别值得一提的是，当 Claude 执行一个脚本时，只有脚本的输出结果（例如“验证通过”或具体的错误信

息）会进入上下文，而脚本代码本身则完全不占用空间。

下面这个表格可以帮助你更直观地理解这套机制：

层级 加载时机 消耗的Token成本 元数据 (Metadata) 始终加载（对话启动时） 每个 Skill 约 100 Tokens 指令 (Instructions) 当 Skill 被触发时 小于 5k Tokens 资源 (Resources) 按需使用或执行时 几乎无上限（因为内容本身不进入上下文）

了解了这套高效的机制后，我们来看看你能实际使用哪些类型的 Skills。

3. Skills 的两种类型：官方“预制菜”与个人“私房菜”

Claude Skills 主要分为两大类，你可以将它们形象地理解为官方提供的“预制菜”和用户自己调制的“私房菜”。

预置 Skills (Pre-built Skills)

定义： 这是由 Anthropic 官方提供、开箱即用的能力包，专注于处理常见的办公和文档任务。它们经过精心优化，确保稳定可靠。

实例列举：

PowerPoint (pptx): 创建、编辑和分析演示文稿。

Excel (xlsx): 创建电子表格、分析数据并生成图表。

Word (docx): 创建、编辑和格式化 Word 文档。

PDF (pdf): 生成格式化的 PDF 文档、提取内容或填写表单。

自定义 Skills (Custom Skills)

定义： 这是用户根据自己特定的需求、工作流程或专业知识创建的个性化能力包。你可以封装任何你希望 Claude 掌握的“独门绝技”。

强调易用性： 创建一个简单的自定义 Skill 并不需要编程知识。你只需要编写一个包含清晰指令的 Markdown 文件 (SKILL.md) 即可。

亮点功能： Anthropic 甚至提供了一个名为 “skill-creator” 的特殊 Skill。你可以通过与 Claude 对话的方式，让它帮你创建、构建和打包一个新的 Skill，这极大地降低了上手的门槛。

拥有了这些强大的“武器”，你可以在哪些“战场”上使用它们呢？

4. 在哪里使用 Skills?

Claude Skills 具备出色的跨平台能力，你可以在 Claude 的主要产品中使用它们。然而，不同平台的使用方式、技术要求和分享范围有显著区别，理解这些差异是高效使用 Skills 的关键。

使用条件：付费计划专属

首先需要明确，Skills 是付费计划用户可用的功能，包括 Pro、Max、Team 和 Enterprise 等订阅方案。

在 claude.ai：通过界面轻松上传

对于网页版和桌面应用用户，使用自定义 Skills 的方式最为直观：

> 操作方式： 用户通过 Settings > Features (设置 > 功能) 菜单上传自定义 Skills。
>
> 文件格式： 自定义 Skills 需要打包成 ZIP 文件 进行上传。
>
> 前置条件： 必须启用 “code execution and file creation” (代码执行与文件创建) 功能。
>
> 分享范围： 在 claude.ai 上传的 Skills 是个人私有的，团队成员无法直接使用，需要各自上传。

通过 Claude API：为应用集成专业能力

对于希望将 Skills 能力集成到自己应用程序中的开发者而言，API 提供了强大的编程接口：

> 操作方式： 在 API 请求中，通过在 container 参数内指定 skill_id 来调用相应的 Skill。
>
> 技术要求： 使用 Skills 功能需要启用三个特定的 beta 请求头 (beta headers)：code-execution-2025-08-25、skills-2025-10-02 和 files-api-2025-04-14。
>
> 分享范围： 通过 API 上传的自定义 Skills 是工作区共享的，工作区内的所有成员都可以访问和使用。

在 Claude Code：本地化的开发者工作流

Claude Code (命令行编程助手) 提供了与开发者工作流深度集成的独特方式：

> 操作方式： Skills 是基于文件系统的，无需任何上传操作。Claude 会自动发现并加载指定目录下的 Skills。
>
> 存储位置： 个人私有的 Skills 存放在 ~/.claude/skills/ 目录；而项目共享的 Skills 则存放在项目根目录下的 .claude/skills/ 目

录。

分享范围： 个人 Skills 仅自己可用。项目级 Skills 可以通过 Git 等版本控制工具与整个团队同步和共享，实现了工作流的无缝协作。

了解了所有基本概念后，让我们回归初心：这一切对你来说，究竟意味着什么？

5. 总结：为什么 Skills 对你很重要？

Claude Skills 的出现，标志着 AI 助手正在经历一次深刻的进化——从一个“通用的问答机器人”，向一个**“可定制的、懂你的专家级合作伙伴”**转变。它不仅仅是一个新功能，更是一种全新的 AI 协作范式。

对你而言，Skills 带来了三个核心的转变：

从“重复解释”到“一次教会” 你不再是 AI 的临时指令下达者，而是它的“导师”。通过创建 Skills，你将自己的专业知识和工作流程沉淀下来，让 AI 能够长期学习并为你所用。

从“通用输出”到“专属品质” 无论是报告的格式、邮件的语气，还是代码的风格，Skills 都能确保 Claude 的产出严格符合你的标准，为你打造具有个人或团队烙印的高品质内容。

从“被动应答”到“主动协作” 当 Claude 能够根据你的意图自动调用正确的“工具”时，它就不再是一个被动的执行者，而是一个能主动思考、协同解决问题的工作伙伴。

现在，是时候开始探索和使用 Claude Skills 了。去创造属于你的第一个 Skill，开启与 AI 协作的全新篇章吧！

高级话题

Claude Skills vs. MCP

Claude Skills 与 Model Context Protocol (MCP) 相比，其核心优势和设计意图主要体现在**令牌效率**、**用户可访问性/设置简易性**以及**任务抽象级别**上。

核心优势和设计意图对比

1. 令牌效率（TOKEN EFFICIENCY）与渐进式披露 (PROGRESSIVE DISCLOSURE)

这是 Claude Skills 相较于 MCP 的一个**核心技术优势**。

特性	Claude Skills	模型上下文协议 (MCP)
设计核心	**渐进式披露 (Progressive Disclosure)**	**协议标准 (Protocol Standard)**
上下文加载	**高度高效**：在会话开始时，Claude **仅**加载所有已安装 Skills 的元数据（名称和描述），这只消耗大约几十个额外令牌。只有当用户请求与特定 Skill 相关时，模型才会通过 Bash 读取完整的指令（`SKILL.md`）和资源文件，将其加载到上下文窗口中。	**效率较低**：MCP 在令牌消耗方面存在局限性。例如，GitHub 的官方 MCP 曾被指出会消耗数万个上下文令牌。
资源文件消耗	Skill 文件夹中的脚本和大型参考文件在被**实际访问**之前不占用上下文令牌。脚本（例如 Python 实用程序）在沙箱环境中执行时，其代码本身不会进入上下文，只有**输出**会消耗令牌。	较高的令牌消耗会减少大模型本身执行有用工作的空间。

简而言之，Skills 的设计意图是通过**仅加载最小必要信息**来保持提示词简洁，从而显著节约令牌成本和时间。

2. 可访问性与设置简易性

Skills 的设计旨在降低门槛，使其不仅适用于开发者，也适用于普通的企业用户。

特性	Claude Skills	模型上下文协议 (MCP)
核心抽象	任务（Task-focused capability packs）	协议/接口标准
设置与编写	编写方式**简单且声明性强。**Skill 本质上是一个包含 Markdown 文件（`SKILL.md`，带有 YAML 前置数据）和可选脚本/资源的文件夹。非技术用户只需描述意图，甚至可以通过 **“skill-creator”** 助手创建 Skill。	设置**复杂。**MCP 是一个完整的协议规范，涉及客户端、服务器、资源和传输方式。它需要开发者/DevOps 团队来编写和操作服务器，设计认证和授权机制。
使用对象	**权力用户和团队**；或任何寻求自动化重复任务的非开发者。	**平台团队**、**企业**和需要构建代理应用程序的**开发者。**
工作流抽象	Skill 是**重复性工作流**的打包，用于捕捉组织知识、最佳实践和特定领域知识。	MCP 旨在充当**集成架构**，用于标准化 AI 应用与内部系统和外部服务（如数据库、GitHub、JIRA）的连接方式。

3. 可移植性和通用性

Skills 的设计意图是实现 **“创建一次，在 Claude 生态中处处可用”。**

> **Skills 的通用性**：Skills 可以在所有 Claude 界面上使用，包括 Claude Apps (Web/桌面)、Claude Code 命令行工具和 Claude API。Skills 具备**可组合性**，模型可以自动识别并组合多个 Skills 来处理复杂的复合任务（例如，结合 Excel 分析 Skill 和 PowerPoint 演示文稿 Skill）。
>
> **MCP 的开放性**：MCP 作为一个**开放协议**，设计上具有最高的供应商中立性和可移植性。同一 MCP 服务器可以被多个客户端/主机复用。

4. 关系总结

Skills 和 MCP 并非相互替代，而是可以协同工作的：

> **Skills** 提供**程序性知识**（如何完成一项特定任务或工作流程）。
>
> **MCP** 提供**工具访问权限**（将 Claude 连接到外部服务和数据源）。
>
> 一个 Claude Skill 可以包含指令，指导 Claude 调用通过 MCP 暴露的工具。例如，Skill 可以包含一个 Bash 脚本，该脚本使用 MCP 工具来提取表格模式。

一些专家认为，鉴于 Skills 在令牌效率、设置简易性和共享方面具有优势，它们在长期内**可能比 MCP 更重要。**

5. **局限性和安全考量**

> **Skills 的局限性**：Skills 是 Claude 独有的，不直接移植到其他模型宿主上。此外，如果 Skill 包含可执行代码（Python 或 Bash 脚本），它们依赖于**代码执行环境**，因此需要启用 Code Execution Tool，并且要求用户**只安装来自受信任来源的 Skills**，以防恶意代码导致数据泄露或系统损坏。
>
> **MCP 的局限性**：需要开发人员的时间来编写和运行服务器，远程服务器会增加网络延迟。

Skills in Context Engineering

Claude Skills 是从Context Engineering中的工作角色的角度出发，提供了一种类似加载function call的更加灵活和高效的方式来加载工作角色（以所需的技能来定义的）。它在MAS中起作用的地方，是根据业务的需要，灵活动态的加载工作角色，从而从角色上层面上隔离相应的Context（包括工具调用，外部知识，以及角色自身的知识），从而减少无效的，多余的context sharing.

参考

what are skills

Claude Skills: The 3 Automation Secrets That Are Making Enterprise Teams Look Like Wizards

分享这篇文章

相关文章推荐

AI Context Engineeri...

这里将收集Context...

Context Engineeri...

Context Engineering Int...

Context Engineering

Context Engineering 是...