

用代码触摸AI时代的脉搏

老程序员踩坑记

主讲人：庄表伟

时间：2025.4




缘起





从使用到深入使用

- 
- OpenAI ChatGPT扑面而来
 - 老范找我聊天，他自己在玩本地AI
 - 与李骏、王伟老师聊天，讨论搭建本地AI的可能性
 - 中国开源年度报告的新闻整理工作：文章-->分类-->摘要-->数据库



尝试写一个开源项目

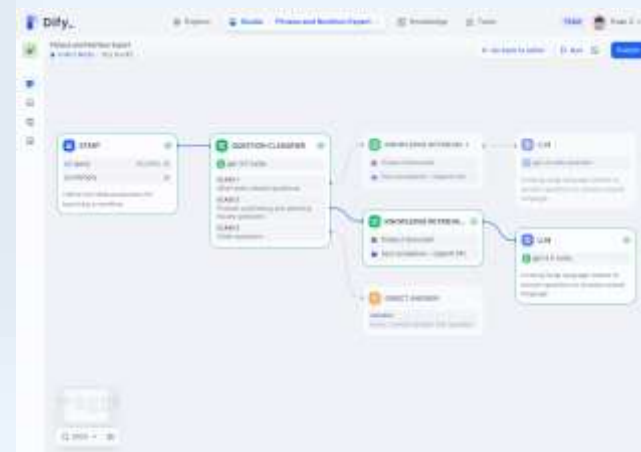
- 尝试其他的AI能力
 - RAG
 - Text2SQL
 - Code Generation
- 最开始的重构
- 与雪愚聊天
 - 从可编程提示词，到SmartPrompt



第一次对外分享：介绍SmartPrompt框架

- Ruby DSL：一种专门为特定问题域设计的编程语言或语法
- 用ERB作为提示词模板：Ruby语言中的一种模板引擎，它允许将Ruby代码嵌入到HTML或其他文本格式中。
- 多模型切换：
- Worker + Workflow

```
workers > get_code.rb
1 SmartPrompt.define_worker :get_code do
2   use "siliconflow"
3   sys_msg "You are a helpful programmer."
4   model "Qwen/Qwen2.5-Coder-7B-Instruct"
5   prompt :generate_code, {
6     name: params[:name],
7     description: params[:description],
8     input: params[:input],
9     output: params[:output]
10  }
11  code_text = safe_send_msg
12  if code_text.include?("Failed to call LLM after")
13    code_text
14  else
15    model "meta-llama/Meta-Llama-3.1-8B-Instruct"
16    prompt :get_code, {code_text: code_text}
17    safe_send_msg
18  end
19 end
```



功能演示：应用的目录结构

- config
 - llm_config.yml
- log
 - log.txt
- templates
 - *.erb
- workers
 - *.rb
- hello.rb
- code.rb
- translat.rb
- embedding.rb
- rag.rb
-

```
# gem install smart_prompt
```

```
config > ! llm_config.yml
1  adapters:
2    openai: OpenAIAdapter
3    ollama: OllamaAdapter
4  logger_file: ./log/log.txt
5  llms:
6    siliconflow:
7      adapter: openai
8      url: https://api.siliconflow.cn/v1/
9      api_key: ENV["APIKey"]
10     default_model: Qwen/Qwen2.5-7B-Instruct
11   llamacpp:
12     adapter: openai
13     url: http://localhost:8080/
14   ollama:
15     adapter: ollama
16     url: http://localhost:11434/
17     default_model: qwen2.5
18 default_llm: siliconflow
19 worker_path: "./workers"
20 template_path: "./templates"
```

功能演示：翻译

```
translate_txt.rb > ...
1  require "../smart_prompt/lib/smart_prompt"
2  engine = SmartPrompt::Engine.new("../config/llm_config.yml")
3  Words = {}
4
5  def translate_text(engine, text)
6    checked = true
7    count = 0
8    result = ""
9    while checked == true && count < 10
10     puts "try #{count}"
11     result = engine.call_worker(:categorized_translation, {
12       text: text,
13       source_language: "English",
14       target_language: "Chinese"})
15     checked =
16       (result.include?("()") && result.include?("(")) ||
17       (result.include?("'") || result.include?("'json")) ||
18       result.include?("**Output**") ||
19       result.include?("**输出**") ||
20       (result.split("\n").size > text.split("\n").size)
21     count += 1
22   end
23   result
24 end
25
26 book = File.read("../OpenLife.txt")
27 new_book = File.new("../OpenLife_new.txt", "w+")
28
29 book.split("\n").each do |line|
30   if line.strip.empty?
31     result = line
32   elsif Words.has_key?(line)
33     result = Words[line]
34   else
35     result = translate_text(engine, line)
36     Words[line] = result
37   end
38   new_book.puts result
39 end
40 new_book.close
```

```
SmartPrompt.define_worker :categorized_translation do
  use "siliconflow"
  model "Qwen/Qwen2.5-Coder-7B-Instruct"
  prompt :cat_trans, {text: params[:text], source_language: params[:source_language]}
  json = send_msg
  model "Qwen/Qwen2.5-72B-Instruct-128K"
  if word?(params[:text])
    prompt :senior_translator, {
      json: json,
      text: params[:text],
      source_language: params[:source_language],
      target_language: params[:target_language]
    }
    result = send_msg
  else
    prompt :senior_translator2, {
      json: json,
      text: params[:text],
      source_language: params[:source_language],
      target_language: params[:target_language]
    }
    result = send_msg
  end
  result
end
```


功能演示：翻译

templates > cat_trans.erb

```
1 # Command:
2 **Input**: <%= source_language %>
3 **Output**: Analysis
4 1. **Category**: Based on the content of a given text, determine to which of the
5     - Abbreviations and acronyms
6     - Dates and times (No translation required)
7     - Numerical value (No translation required)
8     - Monetary units
9     - Units of measurement
10    - Chapter headings
11    - Proper nouns
12    - Literature citations
13    - Fixed expressions, phrases, idioms and expressions
14    - Titles of literary and artistic works
15    - Legal and regulatory provisions
16    - Technical terms
17    - File name (No translation required)
18    - Dialogue, inflections, slang and colloquial expressions
19    - Historical and cultural contexts
20    - Formatted texts, formulas and symbols
21    - URI/URL (No translation required)
22    - Other complex texts
23 2. **Instances**: Under this category, enter actual examples from the text.
24
25 # Example:
26
27 **Input**: NASA's mission was launched on March 1, 2023, at 10:30 PM.
28 **Output**(JSON format):
29 {
30   "category": ["Abbreviations and acronyms", "Dates and times"],
31   "Instances": {
32     "Abbreviations and acronyms": ["NASA (name of organization)"],
33     "Date & times": ["March 1, 2023", "10:30 PM"]
34   }
35 }
36
37 **Input**: <%= text %>
38 **Output**(JSON format):
```

templates > senior_translator.erb


```
1 # Analysis:
2 <%= json %>
3
4 # Command:
5 1. Output only translated content
6 2. Please follow the input file format strictly and do not add anything else.
7 3. Based on the results of the above analysis, translate the following <%= source_language %>, into <%= target_language %>.
8
9 # Example:
10 **Input**: , John Wiley & Sons.
11 **Output**: , 约翰·威利父子.
12
13 # Task:
14 **Input**: <%= text %>
15 **Output**(Plain text):
```

templates > senior_translator2.erb

```
1 # Analysis:
2 <%= json %>
3 # Command:
4 0. The input just one word
5 1. Output only translated content
6 2. Please follow the input file format strictly and do not add anything else.
7 3. Based on the results of the above analysis, translate the following <%= source_language %>, into <%= target_language %>.
8
9 # Example:
10 **Input**: world
11 **Output**: 世界
12
13 # Task:
14 **Input**: <%= text %>
15 **Output**(Plain text):
```


功能演示：抓取新闻

```
workers >  get_news.rb
1  SmartPrompt.define_worker :get_news do
2    url = params[:text]
3    html = call_worker(:download_page, {url: url})
4    text = call_worker(:html_to_text, {html: html})
5    use "siliconflow"
6    sys_msg "You're a journalist familiar with open source-related news coverage."
7    model "Qwen/Qwen2.5-7B-Instruct"
8    prompt :analyzing_news_content, {news: text}
9    news_json = safe_send_msg
10   f = File.open("news.json", "w+")
11   f.puts news_json
12   f.close
13   prompt :generate_sql2, {news: news_json}
14   sql = safe_send_msg
15   sql
16 end
```

```
templates >  analyzing_news_content.erb
```

```
1  Based on the content of the page provided below, the content of N news items
2  was analyzed and presented in the following format:
3
4  ...json
5  [
6    {
7      original_title: "",
8      chinese_title: "",
9      date: "",
10     summary: "",
11     categories: ["", ""]
12   },
13 ]
14 ...
15
16 Input news:
17 <%= news %>
18
19
20
```

功能演示：生成代码

```
code.rb
1 require "../smart_prompt/lib/smart_prompt"
2 engine = SmartPrompt::Engine.new("../config/llm_config.yml")
3 result = engine.call_worker(:get_code, {
4   name: "calculate_triangle_area",
5   description: "calculates the area of a triangle",
6   input: "the base and height of the triangle",
7   output: "the area as a float"
8 })
9
10 if result.include?("Failed to call LLM after")
11   puts result
12 else
13   code_str = result + "\n" + "puts calculate_triangle_area(8, 10)"
14   eval(code_str)
15 end
```

```
workers > get_code.rb
1 SmartPrompt.define_worker :get_code do
2   use "siliconflow"
3   sys_msg "You are a helpful programmer."
4   model "Qwen/Qwen2.5-Coder-7B-Instruct"
5   prompt :generate_code, {
6     name: params[:name],
7     description: params[:description],
8     input: params[:input],
9     output: params[:output]
10  }
11   code_text = safe_send_msg
12   if code_text.include?("Failed to call LLM after")
13     code_text
14   else
15     model "meta-llama/Meta-Llama-3.1-8B-Instruct"
16     prompt :get_code, {code_text: code_text}
17     safe_send_msg
18   end
19 end
```

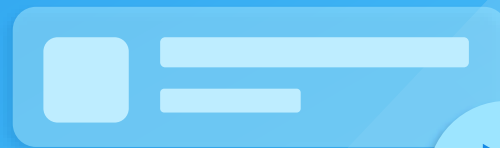
```
tes > generate_code.erb
Write a Ruby function called "<%= name %>" that <%= description %>.
The function should accept <%= input %> as input and return <%= output %>.
Just return the source code to me, no other explanations needed.
```

```
tes > get_code.erb
Remove all non-code text and return only the code. Not even characters like ``.
<%= code_text %>
```

```
def calculate_triangle_area(base, height)
  (base * height) / 2.0
end
puts calculate_triangle_area(8, 10)
40.0
```



产品驱动与架构驱动



从SmartPrompt到SmartAgent

- SmartPrompt的后续发展计划——通通都还没有实现
 - 支持多模态大模型
 - 多个Worker、多个模型并发
 - 对于Worker的输出结果进行调优
 - 系统级Worker：文件管理、数据库管理等
- 实际情况
 - 我想支持逐字显示的效果
 - 一个契机，让我看到了一下DeepSeek的Function Calling
 - 理解回调函数
 - 从SmartPrompt中重构出SmartAgent

让ChatGPT帮我设计SmartAgent

前置说明：我会给你发布清晰的任务说明，我发布的任务分为三类：

- 知识/逻辑类：这类任务的输出结果有对错之分
- 场景/应用类：这类任务的输出结果有好坏之分
- 艺术/创意类：这类任务的输出结果只有美丑之分（而且是因人而异的）

任务类型：场景/应用类

具体任务类型：帮我完善一个系统的架构设计

依赖知识：

- 这是一个Ruby语言框架

任务背景：

- 这个框架用于更好的调用各种大模型
- 输入的是一个大致的设计思路，还比较模糊

任务输入：我想用ruby语言设计一个名叫SmartAgent的框架。大概的思路如下，请给我一些反馈意见，帮助我完善这个设计。

- 用户能够借助这个SmartAgent的框架，完成一些非常复杂的工作。
- 任务首先被分为简单任务和复杂任务。
 - 简单任务分为：适合用大模型完成任务，适合用代码完成的任务。
 - 适合用大模型完成任务，需要借助SmartAgent的框架，判断选择哪一个合适的大模型，选择（或生成）对应的合适的提示词。
 - 适合用代码完成的任务，需要借助SmartAgent的框架，判断是直接调用已经存在的代码，还是下载网络上的某个gem，还是通过LLM生成一段代码。
 - 复杂的任务，需要借助大模型的能力，将任务分解为多个子任务，并以一种适当的DSL，描述这些子任务如何在一起协作。

- 代码类的任务，可以再进行进一步细分

- 网络类任务：HTTP协议访问Web、HTTP协议调用RESTful API
- 数据类任务：查询数据、理解数据结构、操作数据结构，操作数据内容（添加、修改、删除）
- 文件类任务：单个文件的读写、一个文件夹下的多个文件的读写
- 运算类任务：主要是指各种各样的算法（函数）

- 这个系统对于大模型的应用，主要按照输入输出的信息类型来做分类

- 无论输入还是输出，信息类型都包括：文本、数据、图像、视频、音频。
- 例如：输入“请计算1加到10的总和”，系统可以按如下方式处理：自然语言（文本）->生成计算代码（文本）->执行代码得到结果（数据）->自然语言告知结果（文本）

执行思路：请帮我深入分析这个设计，一步一步的考虑各种情况与应用场景，同时考虑可能存在的例外情况，经过你的反思、综合、整理，最后告诉我一个你认为最完善的回答。

输出要求

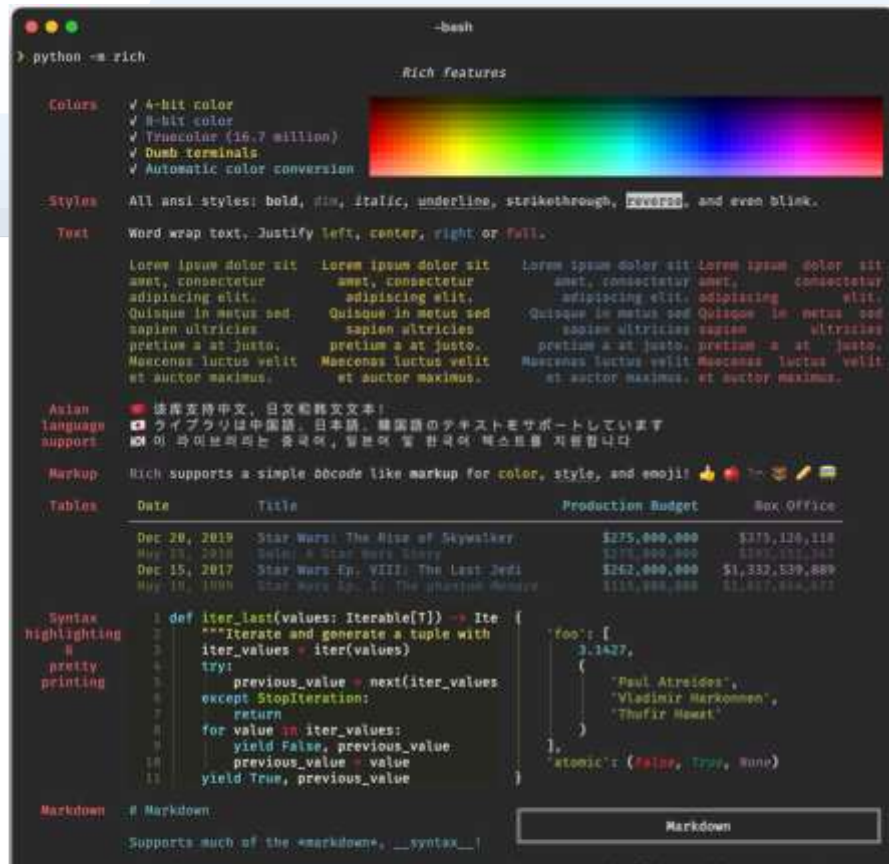
输出要求--内容约束：

- 一份结构清晰、完整、全面的设计稿
- 这个设计稿，还应该加上你认为必要的评注与说明

输出要求--格式约束：markdown

从Rich到RubyRich

- 我真的眼馋Python的生态
 - <https://github.com/Textualize/rich>
 - <https://github.com/Textualize/textual>
- 所以，我想用Ruby实现一个类似的
 - https://github.com/zhuangbiaowei/ruby_rich
- 实际上：勉强糊了一个雏形 :(



```
-bash
> python -m rich

Rich Features

Colors
  ✓ 4-bit color
  ✓ 8-bit color
  ✓ Truecolor (16.7 million)
  ✓ Dumb terminals
  ✓ Automatic color conversion

Styles
  All ansi styles: bold, dim, italic, underline, strikethrough, reverse, and even blink.

Text
  Word wrap text. Justify left, center, right or full.

  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Quisque in metus sed sapien ultricies pretium a at justo.
  Maecenas luctus velit et auctor maximus.

  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Quisque in metus sed sapien ultricies pretium a at justo.
  Maecenas luctus velit et auctor maximus.

  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Quisque in metus sed sapien ultricies pretium a at justo.
  Maecenas luctus velit et auctor maximus.

Asian language support
  本库支持中文、日文和韩文文本！
  ライブラリは中国語、日本語、韓国語のテキストをサポートしています
  이 라이브러리는 중국어, 일본어 및 한국어 텍스트를 지원합니다

Markup
  Rich supports a simple bbcode like markup for color, style, and emoji! 🍌 🍷 🐼 🦉 🦊

Tables
  Date      Title      Production Budget   Box Office
  Dec 20, 2019  Star Wars: The Rise of Skywalker   $275,000,000   $275,120,118
  May 15, 2018  Solo: A Star Wars Story          $275,000,000   $270,910,041
  Dec 15, 2017  Star Wars Ep. VIII: The Last Jedi $262,000,000   $1,332,539,889
  May 19, 2019  Star Wars Ep. IX: The Phantom Menace $113,000,000   $1,017,014,477

Syntax highlighting
  #
  pretty printing

def iter_last(values: Iterable[T]) -> Iterator[T]:
    """Iterate and generate a tuple with iter_values"""
    try:
        previous_value = next(iter_values)
    except StopIteration:
        return
    for value in iter_values:
        yield False, previous_value
        previous_value = value
    yield True, previous_value

Markdown
  # Markdown
  Supports much of the *markdown*, __syntax__!
```



我跟ChatGPT最早的几段交流

<https://github.com/Textualize/rich>

我想要基于rich这个项目，开发一个ruby版本的rich，请给我一些建议。

我打算先开始阶段 1：核心功能

- * 实现基础文本样式（如颜色和加粗）。
- * 提供简单的终端输出功能。

请帮我生成相关的代码。

请根据你的建议，开始下一步扩展：

- 增加更多样式（如斜体、闪烁）。
- 提供更友好的错误提示（例如用户输入的颜色不存在时）。
- 创建一个 Console 类，用于管理多行输出和复杂的终端布局。

帮我生成相关的代码

阶段 2：表格与动态内容
添加表格渲染支持。
实现动态进度条。

请帮我生成相关代码

请帮我，表格功能扩展

支持单元格颜色和样式化（结合 RichText 类）。
允许行高自定义，支持多行单元格。

生成代码

遇到两个bug：

1. 除了headers这一行的text是有颜色的，后面的各行文字，都是没有颜色的。
2. 除了headers这一行的每一个cell，文字左右是一个空格。后面各行的Cell，文字空格不同，导致多了几个空格。

SmartResearch想要实现什么样的效果

SmartResearch - 学思链 (CoLT)，助力研究的最佳工具

Ruby 3.1+ License MIT

通过迭代学习循环进化的AI智能助手

🌟 核心机制：学思链 (CoLT)

与传统思维链 (CoT) 不同，SmartResearch 实现了增强的学习思考循环：

[思考] → [搜索] → [学习] → [存储] → [循环]

这种自我强化的循环可实现持续知识积累和自适应推理。

🚀 主要功能

- 自主学习引擎
 - 网络/本地知识检索
 - 上下文知识分析
 - 基于SQLite的记忆存储
- 交互式命令行界面
 - 实时思考过程可视化
 - 双重反馈机制：
 - 结果验证
 - 知识修正
- 可扩展架构
 - 模块化插件系统
 - 自定义学习策略
 - 多源数据集成

← 左边的这个Readme
也是用AI生成的

SmartBlog，一个中间产品（小玩具）

Smart Blog

AI生成的高质量博客



AI助手

DeepSeek V3

我的文章

📖 工具的发展与人类世界观的演变：从个体到集体的探索

📖 中国老百姓眼中的中美关税大战：民族气节与经济压力的双重考验

📖 博客的发展历史：从个人日志到全球信息共享平台

📖 《源代码》读后感：比尔·盖茨的传奇起点与创业初心

📖 智能代理与MCP：用Ruby构建MCP SDK与Agent框架

📖 我们需要一种新的写作方式

📖 Apache 基金会治理对话 | 专访新任董事 tison

📖 如何使用DeepSeek V3生成高质量博客文章

技术 · 2025年3月25日 · ✨ AI生成

如何使用DeepSeek V3生成高质量博客文章

🕒 8分钟阅读 👁 1,245浏览

在当今内容创作爆炸的时代，AI写作工具正在彻底改变我们创建博客内容的方式。DeepSeek V3作为最新一代的大型语言模型，能够帮助用户快速生成结构完整、内容丰富的博客文章。本文将详细介绍如何利用这一强大工具提升您的内容创作效率。

1. 准备工作

在开始使用DeepSeek V3生成博客之前，您需要做好以下准备：

注册DeepSeek账号并获取API密钥

安装Smart Blog客户端或配置API环境

明确您的博客主题和目标读者群体


2. 生成博客提纲

与DeepSeek V3交互的第一步是生成一个详细的博客提纲。这可以通过以下提示词实现：

请为我的博客生成一个详细的提纲，主题是“如何使用AI工具提升写作效率”。要求包含：

1. 引言部分
2. 至少5个小节，每节有明确的子标题
3. 每个小节下包含3-5个要点
4. 结论部分

目前的成果

```
agents >  smart_bot.rb
1 SmartAgent.define :smart_bot do
2   call_tool = true
3   while call_tool
4     result = call_worker(:smart_bot, params, with_tools: true, with_history: true)
5     if result.call_tools
6       call_tools(result)
7     else
8       call_tool = false
9     end
10  end
11  if result != true
12    result.response
13  else
14    result
15  end
16 end
17
18 SmartAgent.build_agent(:smart_bot, tools: [:search, :get_code], mcp_servers: [:opendigger, :sequentialthinking_tools, :amap])
```

```
agents > tools >  get_code.rb
1 SmartAgent::Tool.define :get_code do
2   param_define :name, "ruby function name", :string
3   param_define :description, "ruby function description", :string
4   param_define :input_params_type, "define of input parameters: (name:type, name:type ... )", :string
5   param_define :output_value_type, "type of return value.", :string
6   param_define :input_params, "input parameters: (value, value ...)", :string
7   if input_params
8     code = call_worker(:get_code, input_params)
9     if input_params["input_params"][0] == "(" && input_params["input_params"][-1] == ")"
10      code += "\n" + input_params["name"] + input_params["input_params"]
11    else
12      code += "\n" + input_params["name"] + "(" + input_params["input_params"] + ")"
13    end
14    "通过生成的代码:\n #{code} \n得到了结果: #{eval(code)}"
15  end
16 end
```

```
SmartAgent::MCPCClient.define :opendigger do
  type :stdio
  command "node ~/open-digger-mcp-server/dist/index.js"
end

SmartAgent::MCPCClient.define :amap do
  type :sse
  url "https://mcp.amap.com/sse?key=xxxxxxxxxxxx"
end
```

目前的成果

```

└─ agents
  └─ mcps
    ├── amap.rb
    ├── opendigger.rb
    ├── postgres.rb
    └─ sequentialthinking_tools.rb
  └─ tools
    ├── get_code.rb
    ├── get_sum.rb
    ├── get_weather.rb
    ├── search.rb
    └─ smart_bot.rb
  └─ bin
    └─ smart_research
  └─ config
  └─ conversations
  └─ db
  └─ lib
  └─ log
  └─ templates
  └─ test
  └─ workers
  └─ .gitignore
  └─ CONTRIBUTING.md
  └─ Gemfile
  └─ Gemfile.lock
  └─ LICENSE
  └─ README_zh.md
  └─ README.md
```

```

[ 聊天窗 ]
User: 华东师大中山北路校区, 附近有哪些美食?
AI Talking:
Call:
MCP Server is 'sequentialthinking_tools', ToolName is 'sequentialthinking_tools'
params is '{"thought"=>"To comprehensively find and organize nearby food options around East China Normal University's Z hongshan North Road campus, we need to first convert the address to coordinates, then perform a nearby search for food-related POIs, and finally gather detailed information about each POI.", "next_thought_needed"=>true, "thought_number"=>1, "total_thoughts"=>3}'
... done
Call tools completion.
Call:
MCP Server is 'amap', ToolName is 'maps_geo'
params is '{"address"=>"华东师大中山北路校区", "city"=>"上海"}'
... done
Call tools completion.
Call:
MCP Server is 'amap', ToolName is 'maps_around_search'
params is '{"keywords"=>"美食", "location"=>"121.486829,31.227667", "radius"=>"1000"}'
... done
Call tools completion.
以下是根据华东师范大学中山北路校区（位于上海市普陀区）附近的美食整理结果，包括餐厅名称、地址、特色及部分照片链接：
---
### 1. **华东师范大学普陀校区河东食堂**
- **地址**：华东师范大学(中山北路3663号)
- **类型**：校园食堂
- **照片**：[点击查看](http://store.is.autonavi.com/showpic/d5941ee6287e25b6e3f38dfdddbcb493)
### 2. **利兴餐厅**
- **地址**：中山北路3671弄172号华师大一村(中山桥下面)
- **类型**：中式餐厅
- **照片**：[点击查看](http://store.is.autonavi.com/showpic/7fd959ddb1fa120f8c49354c9e5c800fa)
### 3. **千里香馄饨王(曹家巷小区店)**
- **地址**：曹家巷小区西1门旁(金沙江路地铁站5号口步行310米)
- **类型**：馄饨小吃
```

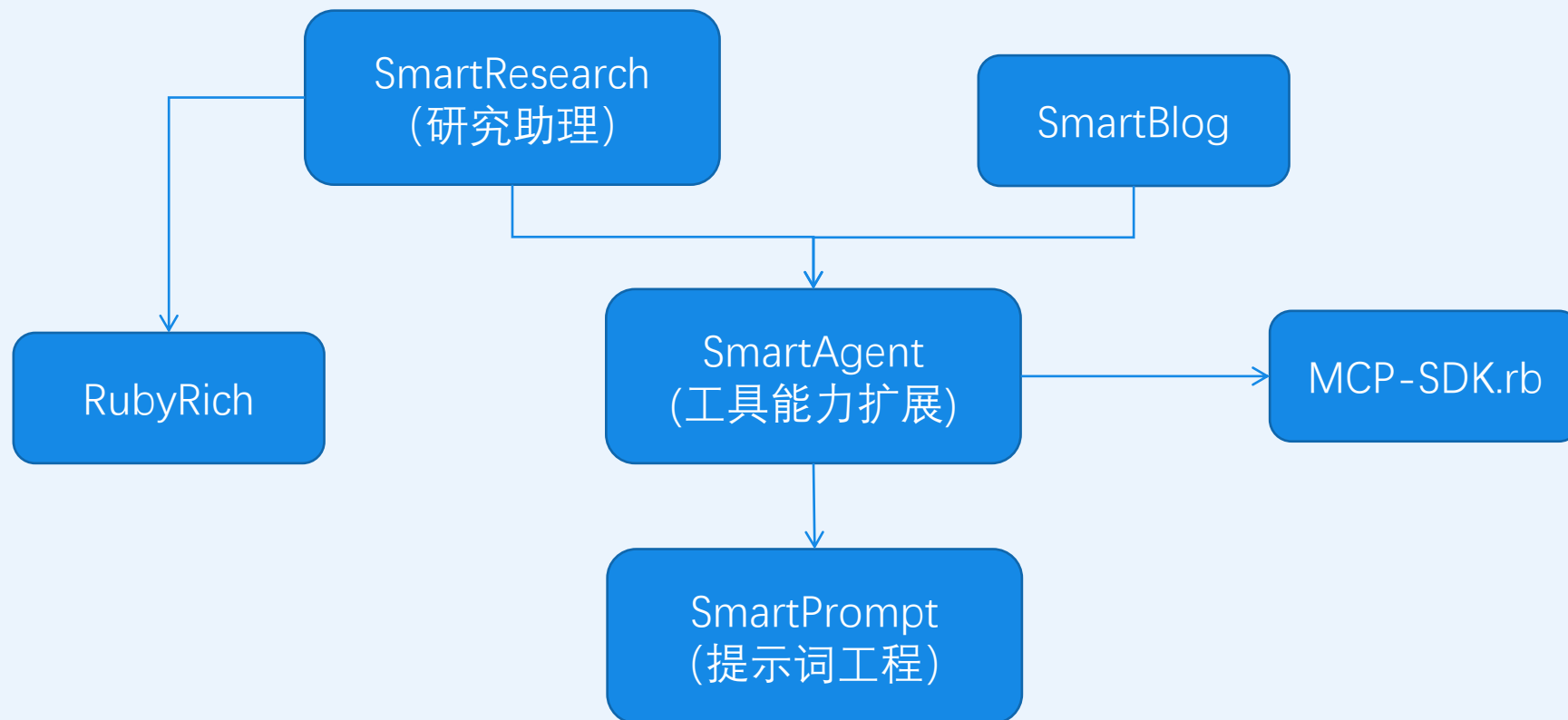
[交流与探索 (F6 = 换行, ↑/↓ = 切换聊天历史) model: Pro/deepseek-ai/DeepSeek-R1]

```
>
```

```

[ 快捷方式 ]
[F1] 帮助
[F2] 交流与探索
[F3] 整理知识库
[F4] 创作与输出
-----
[Ctrl+N] 开启新对话
[Ctrl+D] 删除对话
[Ctrl+S] 保存对话
[Ctrl+O] 加载历史对话
-----
[Ctrl+T] 加载工具或MCP
[Ctrl+W] 切换工作模型
-----
[Ctrl+C] 退出
```

目前的成果

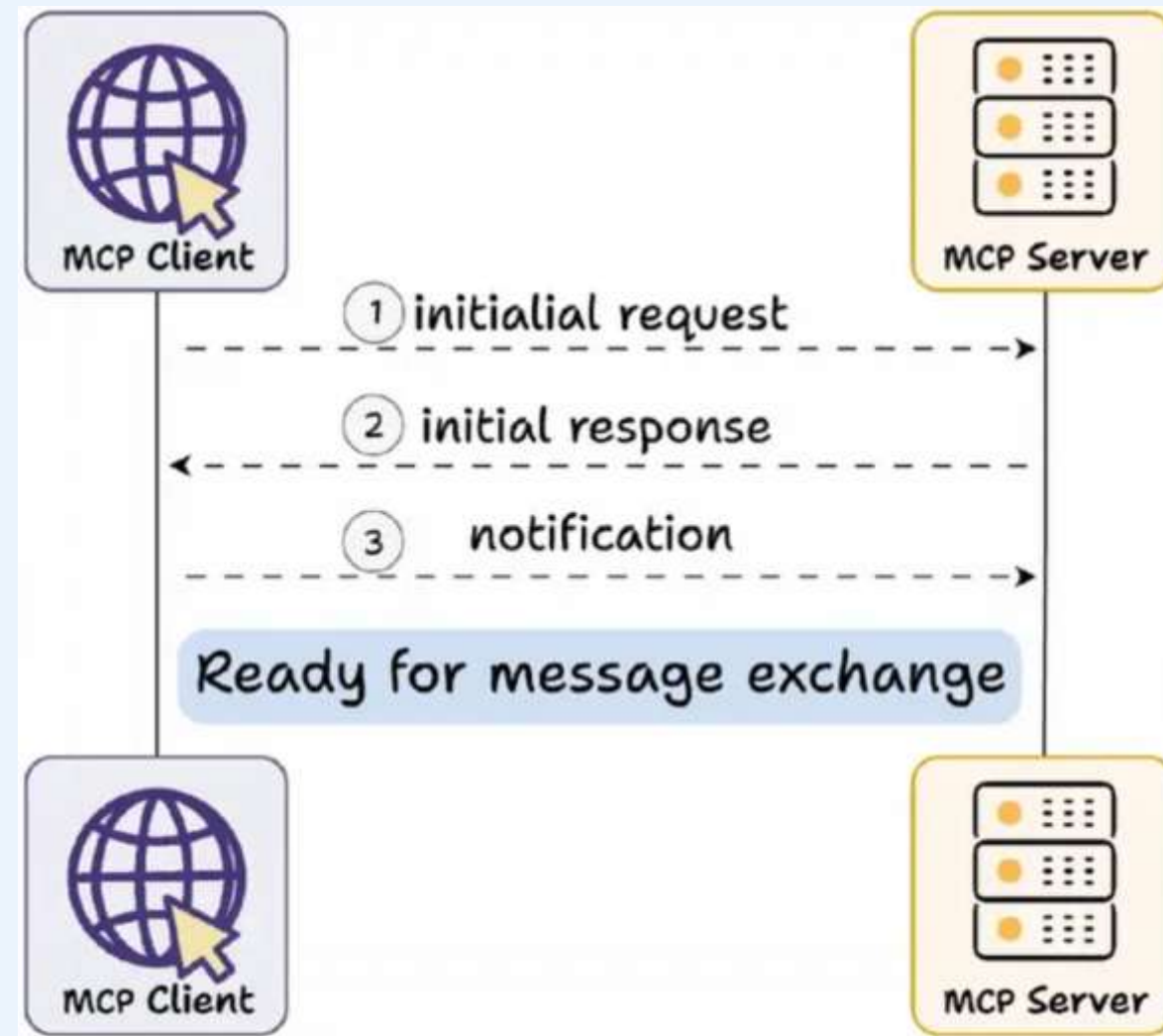
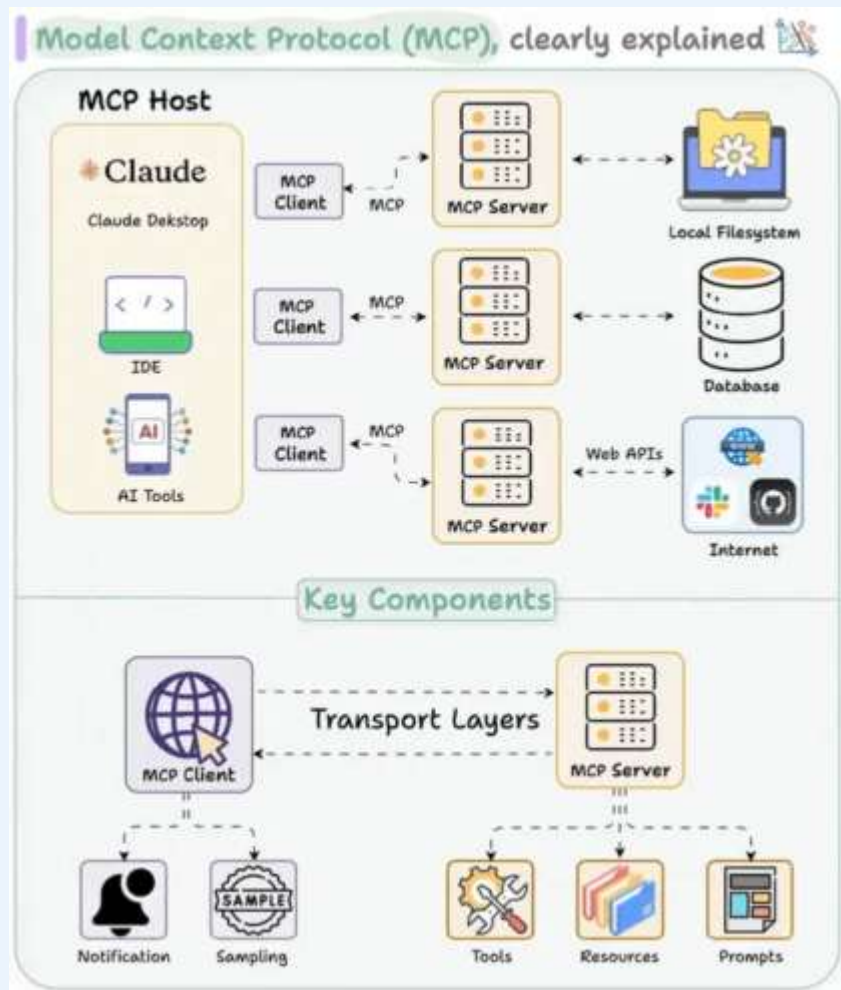




踩坑记



MCP是一个大坑



MCP是一个大坑

- MCP的本质：Function Calling的扩展
 - 再另外加了一些概念：Resources、Prompts、Sampling、Roots（为啥？）
 - 以及保持长链接的双向通讯（为啥？）
- Stdio 与 HTTP SSE
 - Standard Input/Output (stdio), 虽然、但是，还行（我跑通了）
 - Server-Sent Events (SSE), 在Ruby生态里，就没有现成可用的gem
 - <https://github.com/launchdarkly/ruby-eventsource>
 - 不好用
 - <https://github.com/simonx1/ruby-mcp-client>
 - 不好用
- 求助AI帮我写代码，TA写不出来
- 最后的解决方案：借助js，写了一个Proxy

其他小坑与跳坑原因



- AI帮我写了bug，他自己改不掉
 - 尤其是：我只知道现象，不知道原因的时候——他也不知道
- 我太贪心，想让他一口气就帮我写出来一个复杂的功能（甚至整个框架）
- Ruby这门语言，LLM懂得不多
- 看文档不仔细，掉坑里了，才想起来回头去认真看文档
- 难免还是会粗心
- 当然：用AI帮我从坑里跳出来，也很有效
 - 前提是：我自己已经想清楚是什么bug了



我的经验与心得



调试方面的心得

- 写Logger，输出Logger，能够帮助我们更好的发现问题
 - 对于Logger，要有系统性的思考，合理的放置输出的位置
- Debug依然是必备技能
 - 在理解他人的代码时，也很好用
- 把出错信息贴给AI，的确能够快速解决一些常见问题
 - 有时候也不行
- 如何找到更多的bug
 - 从开发者心态，切换到测试者心态

产品方面的心得

- 不懂技术的架构师，不是一个好的产品经理
- 我们需要不断的回头思考：
 - AI是什么？
 - AI能做什么？
 - 我能用AI来做些什么？
 - 以及：我打算用这个产品解决的问题，以前的人们是怎么处理的？
- 以上的AI，可以替换成任何技术
 - 对于技术的能力边界，要具备架构师的理解深度
- 偏好与品味，必不可少
 - 我要做的，首先是我自己想用的产品

架构方面的心得

- **不懂需求的产品经理，不是一个好的架构师**
 - 架构师，其实也是产品经理：他的用户，就是其他的开发者
- 需要不断的思考：
 - 开发者的需求是什么？
 - 开发者会如何使用这个框架？
 - 架构师希望开发者，如何理解与思考？（在某种架构之内）
 - 架构师：做出一组关键技术决策的人
- 命名是非常关键的！
 - Worker 与 Agent
 - Call 与 Please

如何理解大语言模型与AI?

- 使用者的角度、应用开发者的角度、创造者的角度
 - 至少我们这些有能力Coding的人，可以深入到应用层
- 与蚂蚁开源的朋友聊天：如何追踪AI发展的趋势？
 - 概念（名词） --> 技术实现 --> 项目集成
- 我们刚刚走进AI时代的大门
 - 接下来肯定会大不相同
 - 但是，我们也看不到太远
- 我们需要尽快在原有的知识架构上，长出对于新事物的理解
 - 对于AI，写代码可能是一个笨办法，但是却非常牢靠



欢迎提问



谢谢大家

主讲人：庄表伟

时间：2025.4