



Ruby语言的AI实战—— ——从框架到应用

主讲人：庄表伟

时间：2025.07



目录

01 缘起——为什么是Ruby?

02 项目全景图

03 项目演示

04 看看代码

05 我的经验与心得

06 后续的工作计划





01

缘起——为什么是 Ruby?



我的个人经历

工作经历

2006~2009:
印客网

2009~2012:
盛大创新院

2013~2022:
华为

Ruby经历

2006年开始用
Rails

2010年开始用
Redmine做代
码托管

2013年开始给
GitLab添加功
能

开源经历

2010年发起我
们的开源项目

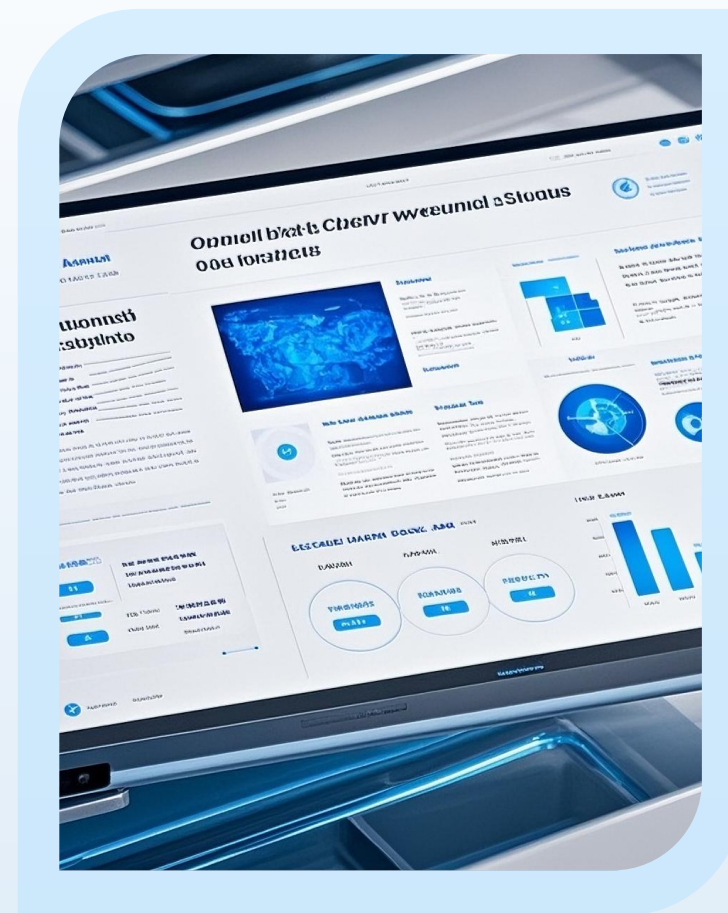
2014年加入开
源社

2023年加入天
工开物开源基
金会

但是，我始终没有正经的用Ruby做过开源项目。

缘起：为何要做这个项目

1. AI大潮扑面而来。
2. 实际的需求：开源大事记，要整理一年几千条新闻。
3. 选择Ruby：“不服气”的起点，“不甘心”的发现。
Python的AI生态那么好，Ruby啥都没有！

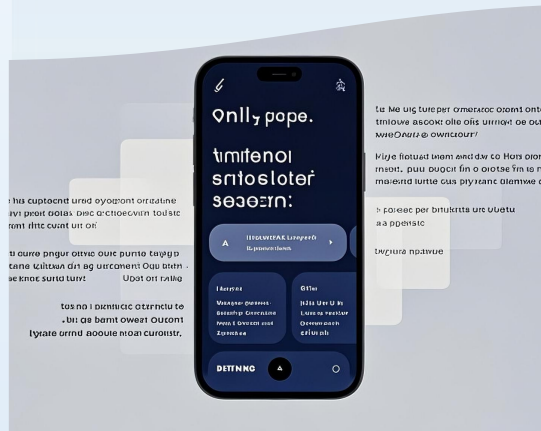


我的开发思路

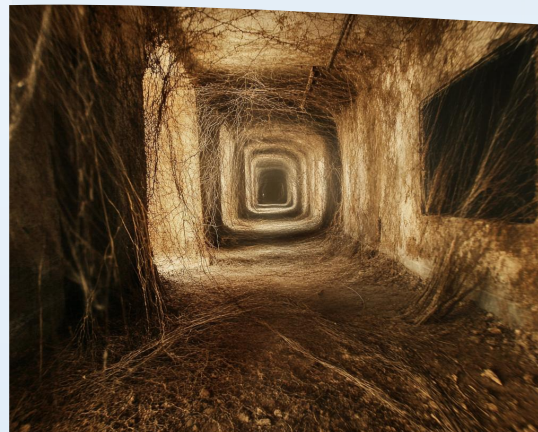
偶尔偷偷瞄一眼
--人家的工作



主要还是
自己闷头干



反复体会与调整，
怎么才是舒服的
DSL?



跟AI聊聊天，但
并不是Vibe
Coding.



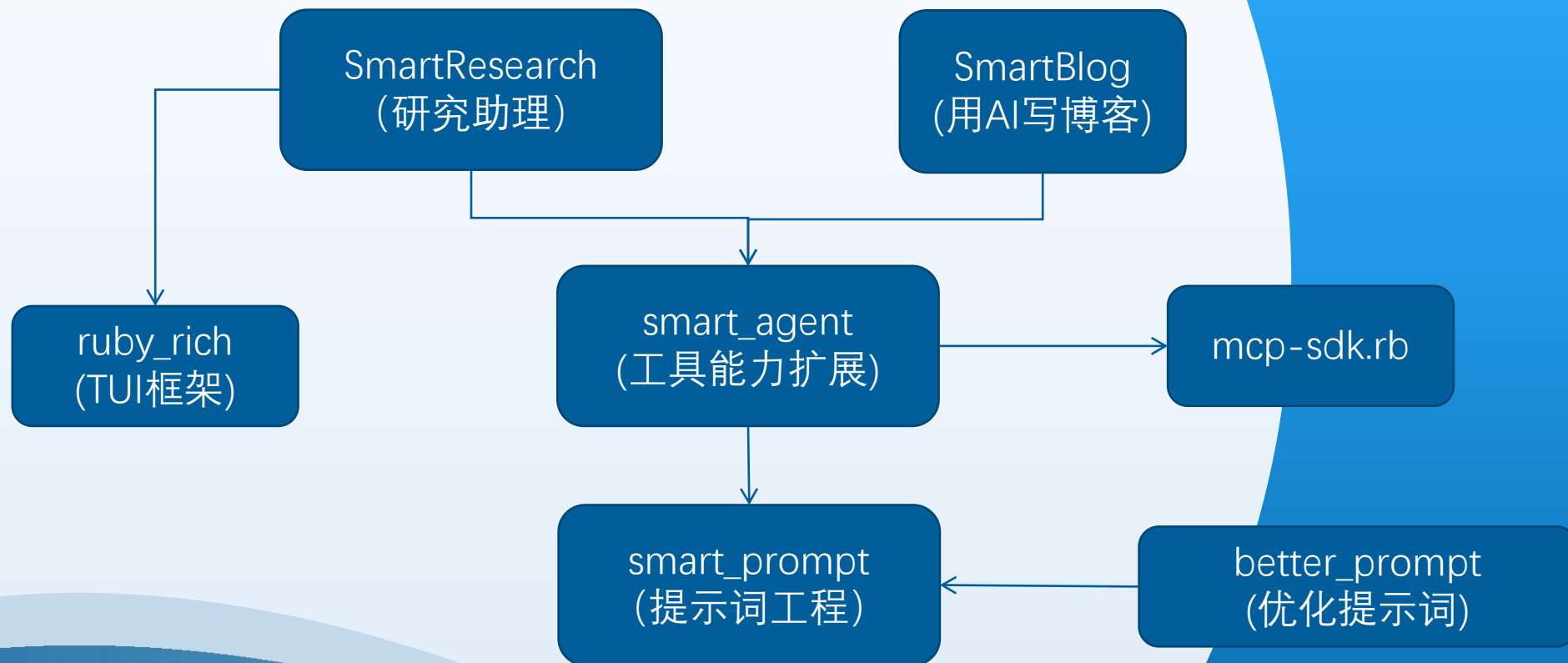


»»» 02

项目全景图



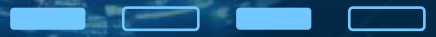
完整架构图



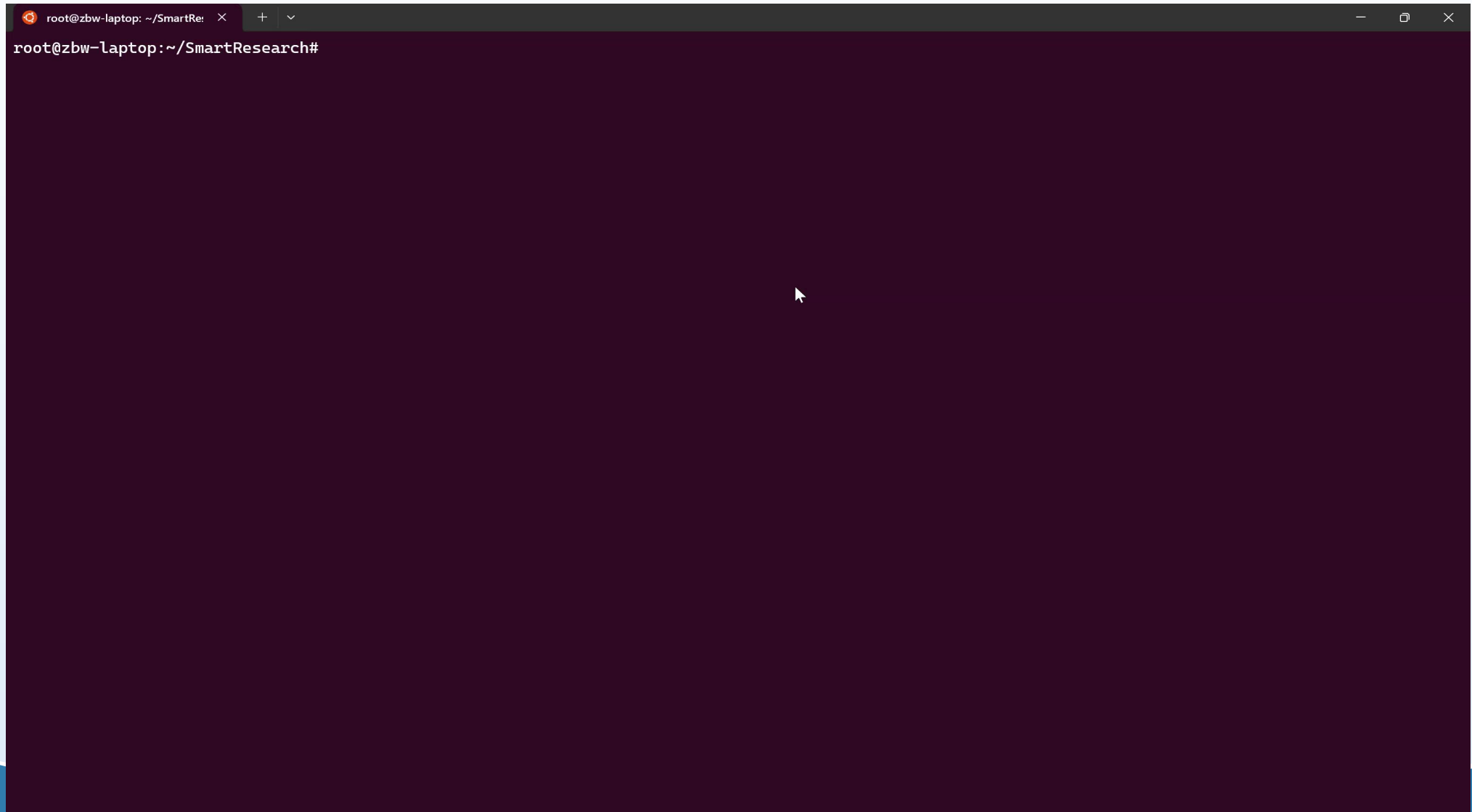


03

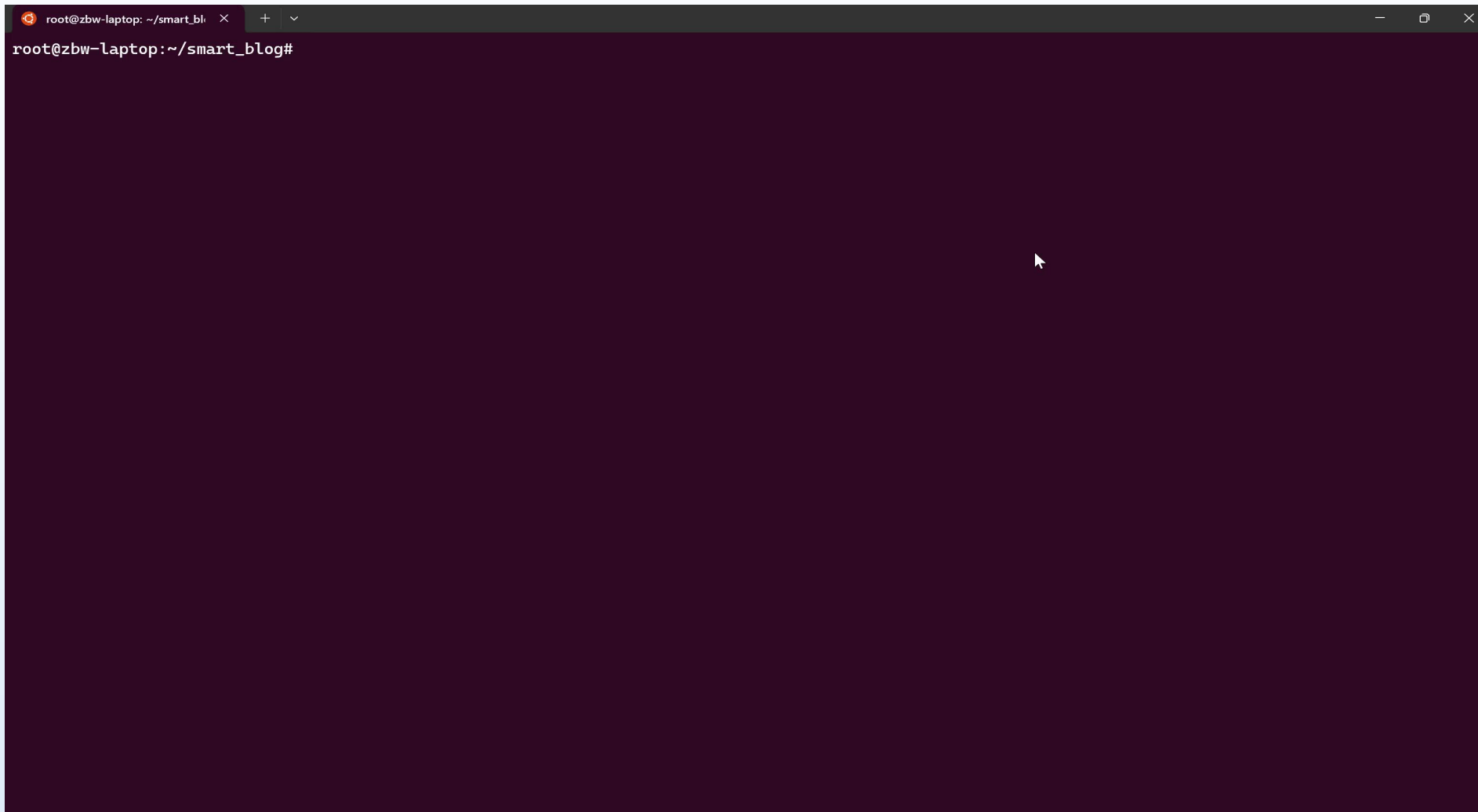
项目演示



SmartResearch演示



SmartBlog演示

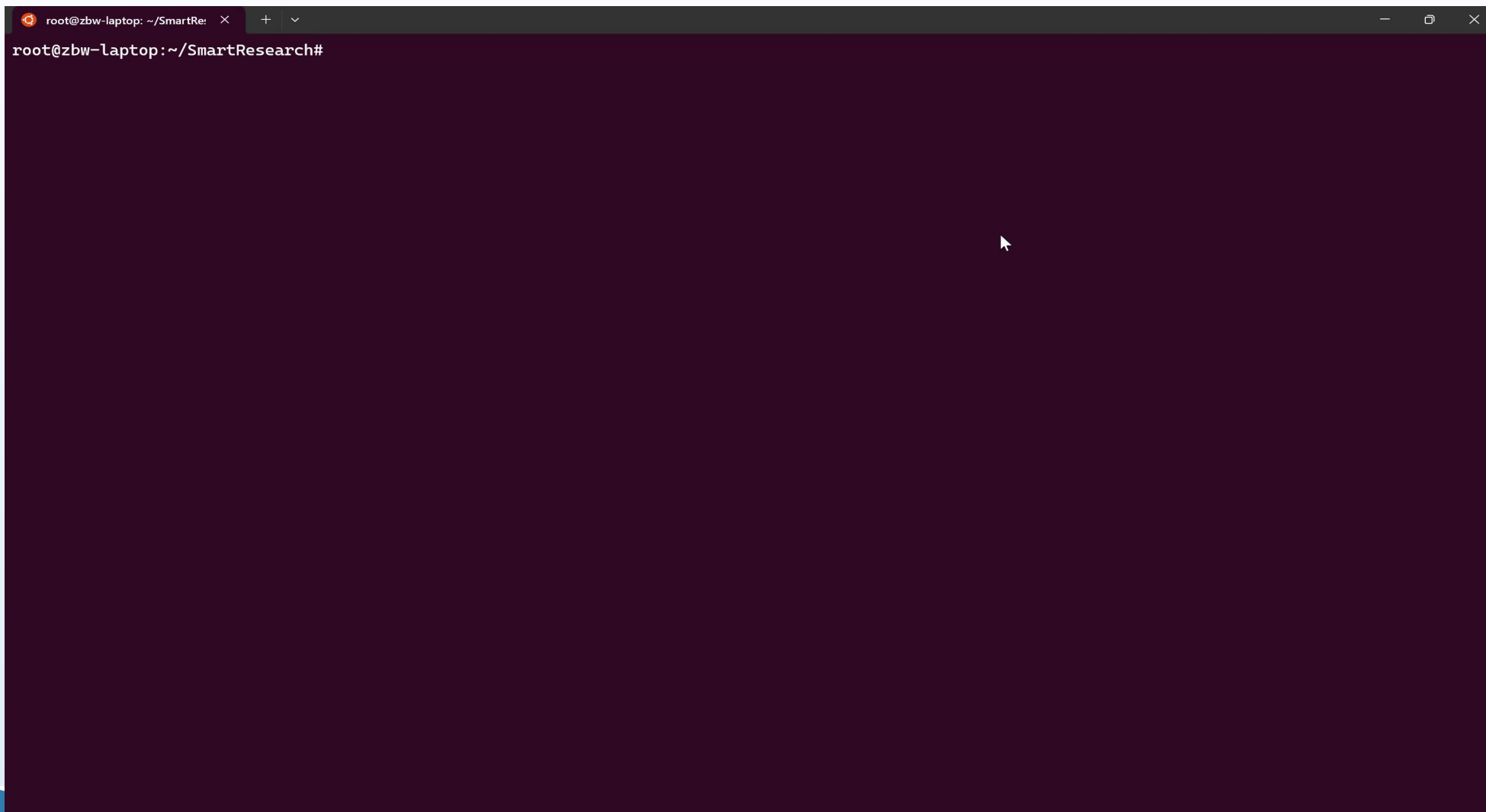


https://zhuangbiaowei.github.io/smart_blog/

SmartBlog演示



BetterPrompt演示





04

看看代码



如何定义一个worker

```
SmartPrompt.define_worker :summary do
  #use "SiliconFlow"
  #model "deepseek-ai/DeepSeek-V3"
  use "deepseek"
  model "deepseek-chat"
  #sys_msg "你是一个强大的整理信息的助手，你主要用中文回答问题。"
  prompt :summarize, { text: params[:text], tool_result: params[:result] }
  send_msg
end
```

```
SmartPrompt.define_worker :daily_report do
  use "ollama"
  model "gemma2"
  system "You are a helpful report writer."
  weather = call_worker(:weather_summary, {location: params[:location], date: "today"})
  prompt :daily_report, {weather: weather, location: params[:location]}
  send_msg
end
```

```
require "smart_prompt"
engine = SmartPrompt::Engine.new("./config/llm_config.yml")
result = engine.call_worker(:daily_report, {location: "Shanghai"})
puts result
```

如何定义一个模板：就是一个ERB文件

```
Please create a brief daily report for <%= location %> based on the following weather information:
```

```
<%= weather %>
```

```
The report should include:
```

1. A summary of the weather
2. Any notable events or conditions
3. Recommendations for residents

如何定义一个agent

```
engine = SmartAgent::Engine.new("./config/agent.yml")

SmartAgent.define :smart_bot do
  call_tool = true
  while call_tool
    result = call_worker(:smart_bot, params, with_tools: true, with_history: true)
    if result.call_tools
      call_tools(result)
      params[:text] = "请继续"
    else
      call_tool = false
    end
  end
  if result != true
    result.response
  else
    result
  end
end

agent = engine.build_agent(:smart_bot, tools: [:get_weather, :search, :get_code], mcp_servers: [:opendigger, :all, :amap])

agent.on_reasoning do |reasoning_content|
  print reasoning_content.dig("choices", 0, "delta", "reasoning_content")
  print "\n" if reasoning_content.dig("choices", 0, "finish_reason") == "stop"
end

agent.on_content do |content|
  print content.dig("choices", 0, "delta", "content")
  print "\n" if content.dig("choices", 0, "finish_reason") == "stop"
end

agent.on_tool_call do |msg|
  puts msg
end

agent.please("请用中文回答，明天上海的天气。")
```

か

```
SmartAgent::Tool.define :get_code do
  desc "Call the LLM to generate a Ruby function with the input details"
  param_define :name, "ruby function name", :string
  param_define :description, "ruby function description", :string
  param_define :input_params_type, "define of input parameters: (name:type, name:type ... )", :string
  param_define :output_value_type, "type of return value.", :string
  param_define :input_params, "input parameters: (value, value ...)", :string
  tool_proc do
    code = call_worker(:get_code, input_params)
    if input_params["input_params"][0] == "(" && input_params["input_params"][-1] == ")"
      code += "\n" + input_params["name"] + input_params["input_params"]
    else
      code += "\n" + input_params["name"] + "(" + input_params["input_params"] + ")"
    end
    "通过生成的代码:\n #{code} \n得到了结果: #{eval(code)}"
  end
end
```

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

SmartAgent::MCPCClient.define :sequentialthinking_tools do
  type :stdio
  command "node /root/mcp-sequentialthinking-tools/dist/index.js"
end

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

SmartAgent::MCPCClient.define :all do
  type :sse
  url "https://mesh-api.dephy.io/mcp/se"
end
```

如何写一个TUI应用

```
✓ lib
  ✓ smart_research
    ✓ components
      change_model.rb
      content.rb
      help_dialog.rb
      input_area.rb
      load_conversation.rb
      main.rb
      root.rb
      save_conversation.rb
      sidebar.rb
    > smart_agent
      application.rb
      learning_loop.rb
      smart_research.rb
```

```
def initialize
  @layout = Component::Root.build
  @layout.split_row(
    Component::Main.build,
    Component::Sidebar.build
  )

  @layout["main"].split_column(
    Component::Content.build,
    Component::InputArea.build
  )

  @engine = SmartPrompt::Engine.new("./config/llm_config.yml")
  @agent_engine = SmartAgent::Engine.new("./config/agent.yml")
  SmartResearch.logger = Logger.new("./log/app.log")
end
```

```
def start
  RubyRich::Live.start(@layout, refresh_rate: 24) do |live|
    live.params[:input_pos] = 2
    live.params[:prompt_list] = [ "> " ]
    live.params[:prompt_no] = 0
    live.params[:current_prompt] = ""
    live.listening = true
    live.app = self
  end
end
```

```
class Sidebar
  def self.build
    layout = RubyRich::Layout.new(name: "sidebar", size: 26)
    draw_view(layout)
    register_event_listener(layout)
    return layout
  end

  def self.draw_view(layout)
    layout.update_content(RubyRich::Panel.new(
      "[F1] 帮助\n" +
      "[F2] 交流与探索\n" +
      "[F3] 整理知识库\n" +
      "[F4] 创作与输出\n" +
      "-----\n" +
      "[Ctrl+N] 开启新对话\n" +
      "[Ctrl+D] 删除对话\n" +
      "[Ctrl+S] 保存对话\n" +
      "[Ctrl+O] 加载历史对话\n" +
      "-----\n" +
      "[Ctrl+T] 加载工具或MCP\n" +
      "[Ctrl+W] 切换工作模型\n" +
      "-----\n" +
      "[Ctrl+C] 退出",
      title: "快捷方式",
      border_style: :green,
    ))
  end
end
```

如何写一个TUI应用

```
def self.register_event_listener(layout)
  layout.key(:f1) { |event, live|
    dialog = HelpDialog.build(live)
    live.layout.show_dialog(dialog)
  }
  layout.key(:ctrl_c) { |event, live| live.stop }
  layout.key(:ctrl_s) { |event, live|
    conversation_name = live.params[:current_conversation_name]
    content_panel = live.find_panel("content")
    unless conversation_name
      name = live.app.get_conversation_name(content_panel.content.split("\n")[0..30].join("\n"))
      name.gsub!("\'", "'")
      conversation_name = name
      live.params[:current_conversation_name] = conversation_name
      content_panel.title = conversation_name
      dialog = SaveConversation.build(conversation_name, live)
      live.layout.show_dialog(dialog)
    else
      dialog = SaveConversation.build("已经保存过了，对话名称为: " + conversation_name, live)
      live.layout.show_dialog(dialog)
    end
    File.open("./conversations/#{conversation_name}.md", "w") do |f|
      f.puts content_panel.content
    end
  }
  layout.key(:ctrl_o) { |event, live|
    files = Dir["./conversations/*.md"]
    dialog = LoadConversation.build(files, live)
    live.layout.show_dialog(dialog)
  }
end
```

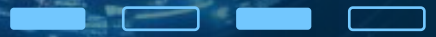
```
class SaveConversation
  def self.build(conversation_name, live)
    width = conversation_name.display_width + 8
    width = 60 if width > 60
    dialog = RubyRich::Dialog.new(title: "保存对话", content: conversation_name, width: width, buttons: [:ok])
    dialog.live = live
    register_event_listener(dialog)
    return dialog
  end

  def self.register_event_listener(dialog)
    dialog.key(:enter){|event, live|
      live.layout.hide_dialog
    }
  end
end
```




05

我的经验与心得



AI能够给我们带来哪些帮助？

我们能够想清楚的部分，AI就能帮我们。我们自己都觉得模糊的，也别指望AI能够直接搞定。

<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。

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

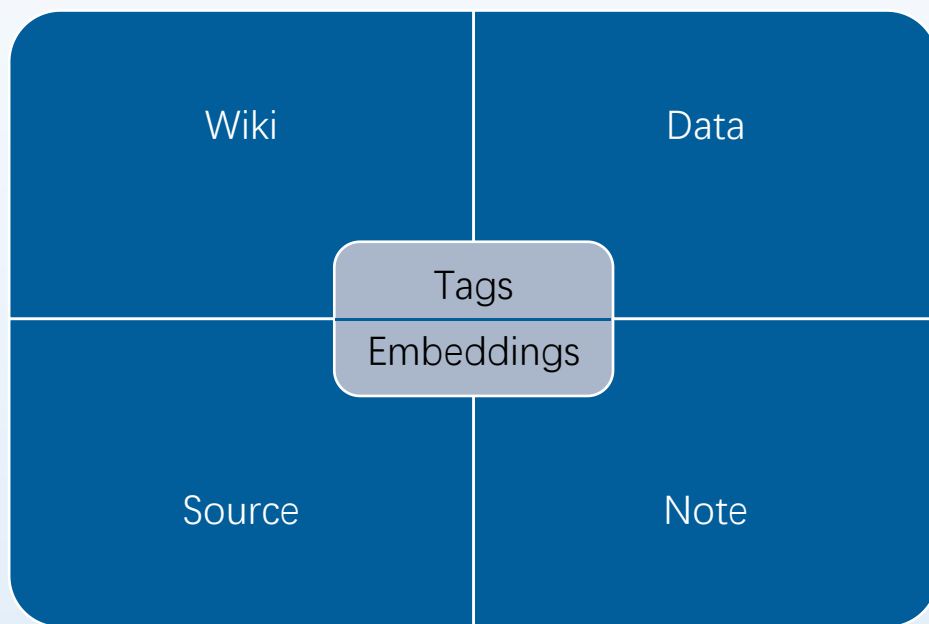


06

后续的工作计划



支持一个好用的基于图和tag的知识库






完成提示词优化框架



- 基于执行效率、调用成本与结果质量，根据不同的需要，选择不同的供应商与不同的大模型。
- 根据用户对于多次调用结果的评分，进一步优化提示词。




RubyRich继续完善

- 
- 更多组件
 - 更好的视觉效果
 - 消除bug




支持多模态大模型

- 
- 向RubyLLM学习，但是不打算简单的集成它的功能
 - DSL可能需要彻底重构





实现Agent2Agent机制

- 
- 一个Agent, 对于另一个Agent来说, 无非是一个可以被调用的MCP而已。
 - 如何实现一个Agent们的聊天室?
 - DSL怎么设计?
 - 还没想好



项目地址

- 
- https://github.com/zhuangbiaowei/smart_prompt
 - https://github.com/zhuangbiaowei/smart_agent
 - <https://github.com/zhuangbiaowei/mcp-sdk.rb>
 - https://github.com/zhuangbiaowei/ruby_rich
 - https://github.com/zhuangbiaowei/better_prompt
 - <https://github.com/zhuangbiaowei/SmartResearch>
 - https://github.com/zhuangbiaowei/smart_blog



欢迎一起来玩!

谢谢大家

主讲人：庄表伟

时间：2025.07

