

ปั้น AI Agent ด้วย DartantAI

คู่มือสร้างเอเจนต์จากพื้นฐาน
สู่การใช้งานจริง

ปั้น AI Agent ด้วย Dartantic AI

คู่มือสร้างเอเจนต์จากพื้นฐานสู่การใช้งานจริง

อนุชิต ชโลธร

คำนำ

หนังสือ **ปั้น AI Agent ด้วย Dartantic AI** เล่มนี้จัดทำขึ้นเพื่อเป็นแนวทางเชิงปฏิบัติสำหรับนักพัฒนาที่ต้องการสร้างและประยุกต์ใช้ระบบปัญญาประดิษฐ์ (AI) ด้วยภาษา Dart ในยุคที่ระบบอัจฉริยะมีบทบาทสำคัญต่อการพัฒนาซอฟต์แวร์ ตั้งแต่การประมวลผลภาษาธรรมชาติ การวิเคราะห์ข้อมูล ไปจนถึงการสร้าง AI Agent ที่สามารถทำงานอัตโนมัติได้อย่างมีประสิทธิภาพ ความเข้าใจและความสามารถในการนำ AI ไปใช้งานจริงจึงกลายเป็นทักษะที่จำเป็นอย่างยิ่ง

แม้ Dart จะเป็นที่รู้จักอย่างแพร่หลายในงานพัฒนาแอปพลิเคชันข้ามแพลตฟอร์ม แต่ด้วยประสิทธิภาพของภาษาระบบนิเวศที่แข็งแกร่ง และความสามารถในการผสานรวมกับบริการและโมเดล AI สมัยใหม่ Dart จึงมีศักยภาพสูงในการพัฒนาระบบอัจฉริยะ หนังสือเล่มนี้มุ่งแสดงให้เห็นว่า Dart และ DartanticAI สามารถนำมาใช้สร้าง AI Agent ได้อย่างเป็นระบบ ยืดหยุ่น และเหมาะสมกับการใช้งานจริง

เนื้อหาภายในเล่มออกแบบในรูปแบบเชิงปฏิบัติ โดยแต่ละบทนำเสนอแนวคิด ขั้นตอน และตัวอย่างโค้ดที่สามารถนำไปใช้งานและ

ต่อยอดได้ทันที ครอบคลุมตั้งแต่พื้นฐานของ AI Agent ไปจนถึง
การเชื่อมต่อกับโมเดลภาษาขนาดใหญ่และการนำระบบไปใช้งานจริง
หวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้จะช่วยให้ผู้อ่านสามารถพัฒนา
โซลูชัน AI ด้วย Dart ได้อย่างมั่นใจและมีประสิทธิภาพ

สารบัญ

แนะนำ Dartantic AI	1
Dartantic AI คืออะไร	1
คุณสมบัติ	2
โครงสร้างพื้นฐานเอเจนต์	4
การติดตั้งและการตั้งค่า	7
การติดตั้งแพ็คเกจ	7
การตั้งค่า API Key	7
ข้อควรระวังด้านความปลอดภัย	8
การสร้างและใช้งานเอเจนต์	9
การระบุโมเดล	11
ผู้ให้บริการโมเดล	12
ชื่อแฝงผู้ให้บริการโมเดล	12
ผู้ให้บริการ	14
ภาพรวมผู้ให้บริการ	14
เปรียบเทียบความสามารถโมเดล	14
การกำหนดค่าผู้ให้บริการ	17

การใช้งานผู้ให้บริการ	18
การตั้งค่าตัวแปรสภาพแวดล้อม	18
การสร้างเอเจนต์	19
คุณสมบัติขั้นสูง	19
การเปิดใช้งานการให้เหตุผล	19
การเรียกใช้เครื่องมือฝังเชิร์ฟเวอร์	20
การจัดการผู้ให้บริการ	20
แสดงรายชื่อผู้ให้บริการทั้งหมด	20
สร้างผู้ให้บริการจากชื่อโมเดลหรือชื่อแฟง	20
การตรวจสอบความสามารถของโมเดล	21
การกำหนดค่าผู้ให้บริการด้วยตนเอง	21
การกำหนด HTTP Headers	22
การตรวจสอบความสามารถของผู้ให้บริการ	22
ประเภทของโมเดล	23
ผลลัพธ์ที่มีโครงสร้าง	23
คำสั่งระดับระบบ	26
การใช้งานคำสั่งระดับระบบ	26
แนวทางการออกแบบ	28
การสนทนา	29

การจัดการประวัติการสนทนา	29
ตัวอย่างการสนทนาพื้นฐาน	30
ตัวอย่างการใช้งานร่วมกับ Streaming	31
รูปแบบการใช้งานขั้นสูง	31
การสนทนาข้ามผู้ให้บริการ	32
การใช้งานร่วมกับเครื่องมือ	32
การแคชพรอมต์	33
การส่งผลลัพธ์แบบสตรีม	35
การสตรีมมิงพื้นฐาน	35
การจัดการ Message History ขณะสตรีม	36
การสตรีมมิงร่วมกับ Tools	37
Extended Thinking	37
การสตรีมผลลัพธ์ที่มีโครงสร้าง	38
การติดตามข้อมูลการใช้งาน	39
ผลลัพธ์ที่มีโครงสร้าง	41
การใช้งานกับ JSON Schema	41
การแปลงผลลัพธ์เป็นอ็อบเจกต์	42
การแปลงเป็น Map	42
การแปลงเป็น Custom Type	43

การสร้าง Schema อัตโนมัติ	44
การใช้งานร่วมกับคุณสมบัติอื่น	45
การสตรีมผลลัพธ์	46
การเรียกใช้เครื่องมือ	46
การรองรับเครื่องมือจากผู้ให้บริการ	47
การเรียกใช้งานเครื่องมือ	49
องค์ประกอบของ Tool	49
ตัวอย่างการใช้งาน	50
รูปแบบการใช้งานขั้นสูง	51
การใช้เครื่องมือหลายตัว	51
การใช้เครื่องมือแบบสตรีมมิง	52
การจัดการข้อผิดพลาด	52
การสร้าง Schema อัตโนมัติ	53
พฤติกรรมเชิงเอเจนต์	56
การให้เหตุผลหลายขั้นตอน	56
เวิร์กโฟลว์ที่ซับซ้อน	58
พฤติกรรมแบบวนซ้ำ	59
อินพุตแบบมัลติมีเดีย	61
รูปแบบการแนบข้อมูล	61

การแนบไฟล์จาก Local Path	61
การแนบไฟล์จาก URL	62
การส่งข้อมูลแบบ Bytes	63
การผสมผสานหลายรูปแบบ	63
การใช้งานขั้นสูง	64
การถอดเสียง (Transcription)	64
OCR (Optical Character Recognition)	65
ผลลัพธ์สื่อมัลติมีเดีย	67
การรองรับการทำงานกับรูปภาพและไฟล์เอกสาร	67
การสร้างสื่อพื้นฐาน	68
การแก้ไขรูปภาพ	69
การสร้างสื่อแบบสตรีมมิง	70
การกำหนดค่าขั้นสูง	72
การระบุ Media Model	72
ตัวเลือกเฉพาะของผู้ให้บริการ	72
การฝังเวกเตอร์	74
การสร้าง Embeddings	74
การวัดความคล้ายคลึงเชิงความหมาย	76
ตัวอย่างการค้นหาเชิงความหมาย	77

การกำหนดค่าขั้นสูง	78
ความไม่เข้ากันของเวกเตอร์	79
การติดตามปริมาณการใช้งาน	80
การติดตาม Usage สำหรับ Chat	80
การติดตาม Usage สำหรับ Embeddings	81
การลองใหม่โดยอัตโนมัติ	83
การทำงานอัตโนมัติแบบไม่ต้องตั้งค่า	83
กลยุทธ์การหน่วงเวลา	84
การบันทึกเหตุการณ์	86
การเปิดใช้งานการบันทึกเหตุการณ์	86
การกำหนดระดับการบันทึกเหตุการณ์	87
ระดับการบันทึกเหตุการณ์	88
การกรองเหตุการณ์ที่บันทึก	88
การกำหนดค่าผ่านตัวแปรสภาพแวดล้อม	89
การเรียกใช้กระบวนการให้เหตุผล	91
การเปิดใช้งานและการเข้าถึงข้อมูล	91
การกำหนดค่าสำหรับแต่ละ Provider	92
OpenAI Responses	93
Anthropic	93

Google	94
ข้อควรทราบ	95
การสตรีมข้อมูล Thinking	95
ผู้ให้บริการแบบกำหนดเอง	97
Dependencies	97
การสร้าง Custom Provider	98
การสร้าง Custom Model	100
การใช้งาน Custom Provider	101
การใช้งานโดยตรง	102
การลงทะเบียนแบบไดนามิก	102
เครื่องมือฝั่งเซิร์ฟเวอร์	104
การรองรับเครื่องมือฝั่งเซิร์ฟเวอร์ของผู้ให้บริการ	104
การเปิดใช้งานเครื่องมือฝั่งเซิร์ฟเวอร์	105
OpenAI Responses	105
Google	106
Anthropic	106
เมทาดาตาแบบสตรีมมิงของเครื่องมือ	107
การรับผลลัพธ์สุดท้าย	108
การกำหนดค่าขั้นสูง	109

เครื่องมือฝังเซิร์ฟเวอร์ของ OpenAI	111
OpenAI Web Search	111
การเปิดใช้งาน	111
การติดตามสถานะผ่าน Metadata	112
การกำหนดค่าเพิ่มเติม	113
OpenAI File Search	113
การเปิดใช้งาน	114
การติดตามสถานะผ่าน Metadata	114
OpenAI Code Interpreter	115
การเปิดใช้งาน	115
การติดตามสถานะผ่าน Metadata	116
การใช้ Container ชั่ว	117
การรับผลลัพธ์ประเภทไฟล์	117
OpenAI Image Generation	118
การเปิดใช้งาน	119
การรับภาพ Preview ระหว่างการสตรีม	119
การรับภาพผลลัพธ์สุดท้าย	120
เครื่องมือฝังเซิร์ฟเวอร์ของ Google	122
Google Search	122

การเปิดใช้งาน Google Search	123
การใช้งาน Google Search	123
การใช้งาน Grounding Metadata	124
Google Code Execution	125
การเปิดใช้งาน Code Execution	125
การทำงานแบบหลายขั้นตอน	126
การรับไฟล์ผลลัพธ์	127
การใช้งานเครื่องมือร่วมกัน	127
เครื่องมือฝั่งเซิร์ฟเวอร์ของ Anthropic	129
Anthropic Web Search	129
การเปิดใช้งาน Web Search	130
การติดตามสถานะผ่าน Metadata	130
Anthropic Web Fetch	131
การเปิดใช้งาน Web Fetch	131
ผลลัพธ์ประเภทไฟล์	132
Anthropic Code Interpreter	133
การเปิดใช้งาน Code Interpreter	133
การทำงานแบบหลายขั้นตอน	133
ผลลัพธ์ประเภทไฟล์	135

การติดตามสถานะผ่าน Metadata	135
การทำงานร่วมกับระบบอื่น	137
การจัดการ Environment	137
การตั้งค่า API Keys	137
ลำดับความสำคัญของตัวแปรสภาพแวดล้อม	139
การเชื่อมต่อ MCP Server	140
Remote MCP Server	140
Local MCP Server	141
การเรียกใช้งานเครื่องมือจากหลายแหล่ง	141
การใช้งาน DotPrompt	142
การสร้างและใช้งาน DotPrompt	142
การส่งตัวแปรไปยัง Template	143
การเรียกใช้ Prompt จากไฟล์	144
ตัวอย่างแอปถามตอบกับ GenAI	146
การสร้างโปรเจกต์	146
การสร้าง Model สำหรับเก็บผลลัพธ์	147
การสร้าง Endpoint	148
การสร้างแอปสำหรับถามคำถามกับ AI	150
การทดสอบการทำงาน	155

ตัวอย่างแอป RAG Agent	160
การสร้างโปรเจกต์	160
การสร้าง Model สำหรับจัดเก็บข้อมูลเวกเตอร์	161
การแปลงเอกสารเป็นเวกเตอร์	162
การสร้าง Endpoint	169
การสร้างแอปสำหรับถามคำถามกับ AI	175
การทดสอบการทำงาน	181
ก้าวต่อไป	185
ดาวน์โหลดซอร์สโค้ด	187

แนะนำ Dartantic AI

การพัฒนาแอปพลิเคชันที่ขับเคลื่อนด้วยปัญญาประดิษฐ์จำเป็นต้องมีเฟรมเวิร์กที่สามารถจัดการความซับซ้อนในการทำงานร่วมกับโมเดลภาษาขนาดใหญ่ (Large Language Models: LLMs) อย่างมีประสิทธิภาพ

Dartantic AI คือเฟรมเวิร์กสำหรับภาษา Dart ออกแบบมาเพื่อวัตถุประสงค์นี้โดยเฉพาะ โดยใช้สถาปัตยกรรมเชิงเอเจนต์ (Agent-based Architecture) เพื่อลดความซับซ้อนในการจัดการผู้ให้บริการ การเรียกใช้เครื่องมือ และการควบคุมบริบทการสนทนา

เนื้อหาในบทนี้จะแนะนำแนวคิดหลัก คุณสมบัติ และโครงสร้างพื้นฐานของ Dartantic AI เพื่อเป็นรากฐานสำหรับการพัฒนาเอเจนต์ขั้นสูงในบทต่อ ๆ ไป

Dartantic AI คืออะไร

Dartantic AI คือ Agent Framework สำหรับภาษา Dart ที่ออกแบบมาเพื่อลดความซับซ้อนในการสร้างแอปพลิเคชัน Generative AI ทั้งฝั่งไคลเอนต์และเซิร์ฟเวอร์ โดยมีเป้าหมายหลัก

เพื่อให้ นักพัฒนาสามารถมุ่งเน้นไปที่การออกแบบพฤติกรรม
อัจฉริยะ (Agentic Behavior) และตรรกะทางธุรกิจ

Dartastic AI มีอินเทอร์เฟซที่เป็นมาตรฐานช่วยจัดการกับความ
แตกต่างของ API ของผู้ให้บริการ ลดความซับซ้อนในการจัดการ
API ที่แตกต่างกัน การแปลงข้อมูล และการควบคุมลำดับการทำงานของ
เอเจนต์ ทำให้กระบวนการพัฒนารวดเร็วและเป็นระบบมากขึ้น

คุณสมบัติ

- **พฤติกรรมเชิงเอเจนต์และการเรียกใช้เครื่องมือหลายขั้นตอน (Multi-step Tool Calling)** เอเจนต์สามารถเรียกใช้เครื่องมือหลายชนิดต่อเนื่องกันโดยอัตโนมัติ เพื่อแก้ไขปัญหาที่ซับซ้อนโดยไม่ต้องอาศัยการสั่งงานจากผู้ใช้ในแต่ละขั้นตอน
- **รองรับผู้ให้บริการหลายราย (Multi-provider Support)** มีระบบรองรับผู้ให้บริการ Generative AI ชื่อนำมาพร้อมใช้งาน (Built-in) เช่น OpenAI, Google, Anthropic, Mistral, Cohere และ Ollama
- **มาตรฐาน OpenAI-compatible API** สามารถเชื่อมต่อกับผู้ให้บริการที่รองรับ OpenAI API ซึ่งเป็นมาตรฐานที่แพร่หลายในปัจจุบันได้อย่างสะดวก

- **การส่งผลลัพธ์แบบสตรีมมิง (Streaming Output)** รองรับการสร้างคำตอบแบบเรียลไทม์ เหมาะสำหรับแอปพลิเคชันที่ต้องการการตอบสนองอย่างต่อเนื่อง เช่น แชทบอต
- **ผลลัพธ์แบบมีชนิดข้อมูล (Typed Outputs)** ผสานระบบชนิดข้อมูลของภาษา Dart เข้ากับการแปลงข้อมูล JSON เพื่อให้ได้ผลลัพธ์ที่มีโครงสร้างชัดเจนและลดความผิดพลาด
- **อินพุตแบบมัลติมีเดีย (Multimedia Inputs)** สามารถประมวลผลอินพุตที่ประกอบด้วยข้อความ รูปภาพ และไฟล์ข้อมูลภายในคำสั่งเดียวกัน
- **การสร้างสื่อ (Media Generation)** รองรับการสร้างและสตรีมรูปภาพ, PDF และไฟล์ประเภทอื่น ๆ จากผู้ให้บริการอย่าง OpenAI, Google และ Anthropic
- **Embeddings** รองรับการสร้างเวกเตอร์สำหรับงานค้นหาเชิงความหมาย (Semantic Search) และการวิเคราะห์ความคล้ายคลึง
- **การแสดงผลกระบวนการคิดของโมเดล (Model Reasoning)** รองรับการแสดงผลกระบวนการให้เหตุผลของโมเดล (ที่มักเรียกว่า “Thinking”) จากผู้ให้บริการที่รองรับคุณสมบัตินี้

- **เครื่องมือฝั่งเซิร์ฟเวอร์ (Server-side Tools)** สามารถเรียกใช้เครื่องมือที่โฮสต์โดยผู้ให้บริการ เช่น การค้นหาเว็บ, การรันโค้ด, และการสร้างภาพ
- **รองรับ Model Context Protocol (MCP)** สามารถทำงานร่วมกับ MCP Server เพื่อขยายขีดความสามารถของเอเจนต์ได้อย่างยืดหยุ่น
- **พร้อมสำหรับใช้งานจริง (Production Ready)** มีระบบบันทึกข้อมูลเหตุการณ์ (Logging), การจัดการข้อผิดพลาด, และกลไกการลองใหม่ (Retry) ที่เหมาะสมสำหรับสภาพแวดล้อมโปรดักชัน
- **ขยายความสามารถได้ (Extensible)** โครงสร้างออกแบบมาให้สามารถเพิ่มผู้ให้บริการ, สร้างเครื่องมือเฉพาะทาง, หรือเชื่อมต่อกับ MCP Server ได้ง่าย

โครงสร้างพื้นฐานเอเจนต์

โครงสร้างพื้นฐานเอเจนต์ (Agent) เป็นจุดเริ่มต้นสำคัญในการพัฒนาแอปพลิเคชัน AI ด้วย Dartastic AI โดยแนวคิดหลักคือการห่อหุ้มความสามารถของโมเดลภาษา (LLM) ไว้ในอ็อบเจกต์เดียว เพื่อให้ผู้พัฒนาสามารถสื่อสารกับโมเดลได้อย่างเป็นระบบ เรียบง่าย และเปลี่ยนผู้ให้บริการได้โดยไม่กระทบกับโค้ดส่วนอื่นของระบบ

```
import 'package:dartantic_ai/dartantic_ai.dart';

final agent = Agent('claude'); // or 'openai',
'gemini', etc.
final result = await agent.send('Hello!');
print(result.output);
```

จากโค้ดข้างต้น นำเข้าแพ็คเกจ dartantic_ai ซึ่งเป็นไลบรารีหลักที่ใช้สำหรับสร้างและจัดการเอเจนต์ AI ในภาษา Dart

```
import 'package:dartantic_ai/dartantic_ai.dart';
```

สร้างอินสแตนซ์ของ Agent โดยกำหนดชื่อผู้ให้บริการโมเดลภาษา (Provider) เป็น 'claude' ซึ่งสามารถเปลี่ยนเป็น 'openai', 'gemini' หรือผู้ให้บริการอื่นที่รองรับได้ โดยไม่ต้องแก้ไขโครงสร้างโค้ดหลัก

```
final agent = Agent('claude'); // หรือ 'openai',
'gemini', etc.
```

ส่งข้อความไปยังเอเจนต์เพื่อให้โมเดลประมวลผล คำสั่งนี้เป็นแบบ asynchronous และจะคืนค่าเป็นอ็อบเจกต์ผลลัพธ์ (ChatResult) ที่บรรจุข้อมูลการตอบกลับจากโมเดล