Agentic Coding

Recommendations

정도현 · 2025-06-26

Agenda

1. Why Agentic Coding?
2. The Basics
3. Language Choice
4. Tools, Tools, Tools
5. Speed & Stability
6. Simplicity First
7. Parallelization
8. Refactoring Mindset
9. Key Takeaways

The Basics

- **권한 확인 비활성화** → 빠른 루프
- **MCP는 최소화**
  - 일반 쉘/스크립트로 충분하면 그대로 사용
- **도커 샌드박스 권장** (위험 완화)
- **IDE 의존 최소화** → Vim + 최종 편집

Language Choice

> “Go is sloppy — perfect for agents.” – Armin Ronacher

- **Go 추천**
  - `context` 패턴이 명시적
  - 테스트 캐싱으로 빠른 루프
  - 구조적 인터페이스 → LLM 친화
  - 생태계 변화 적음
- Python/Rust는 **해석·빌드 지연**과 **매직**으로 난도 ↑

Tools, Tools, Tools

- **무엇이든 도구** (스크립트·로그·MCP 서버)
- 핵심 4원칙
    i. 빠를 것
    ii. 명확한 오류 메시지
    iii. 실패(크래시)는 OK, **행(hang)은 NO**
    iv. 관찰성·디버그 용이
- `Makefile` 활용:
    - `make dev`, `make tail-log` 예시
    - **PID 잠금**으로 중복 기동 방지

Speed Matters

- 추론 비용 + 툴 지연 = 생산성 저하
- **컴파일·부트 시간 최소화**
  - 경량 스크립트 → 3 ms
  - 무거운 서비스 재시작 → 60 s ❌
- 필요 시 **핫 리로드용 데몬** 작성
- 로그는 **간결 + 유의미 + 토글 가능**

Stability & Copy/Paste

- **의존성 업그레이드**는 보수적으로
  - LLM breadcrumb / 주석과 충돌 방지
- “Build It Yourself” – 라이브러리보다 **직접 구현** 선호
- 고정된 생태계 = 예측 가능한 코드 생성

Write Simple Code

- “Dumbest possible thing that works”
- 긴 이름의 **함수 중심** 코드
- 상속 피하기, **ORM 보다 SQL**
- **권한 체크**는 코드 근처에! (AI가 인지하기 쉽게)

Make It Parallelizable

- 에이전트 **단일 속도는 느림** → 다중 인스턴스
- 상태 격리: 별도 체크아웃·DB·Redis 샤드
- `container-use` 등 MCP로 컨테이너 격리 실험

Learn to Refactor

- **타이밍**이 관건
  - 너무 이르면 낭비, 늦으면 LLM 혼란
- 복잡도 ↑ → 컴포넌트 라이브러리 분리 시점
- 리팩터 단계에도 **에이전트** 적극 활용

Key Takeaways

1. **속도**: 짧은 피드백 루프
2. **단순성**: Go, 간단한 도구, 명료한 코드
3. **관찰성**: 로그 + PID 보호
4. **안전 병렬화**: 컨테이너 격리
5. **적시 리팩터**로 복잡도 통제

References

- Armin Ronacher, “Agentic Coding Recommendations” (2025-06-12) https://lucumr.pocoo.org/2025/6/12/agentic-coding/
- 기타: Playwright-MCP, container-use, Cursor Background Agents

Q&A

감사합니다!