TOOLING STRATEGIES (MIXED ENVIRONMENT)

Module 4 — Harwell Prompt Engineering

LEARNING OBJECTIVES

By the end of this module you will be able to:

- Apply the “sidecar” workflow: using browser-based AI effectively alongside an IDE without native AI plugins
- Describe the “integrated” workflow: in-IDE features (context awareness, diff views) for teams using JetBrains or VS Code
- Compare chat interfaces vs. inline code completion: pros, cons, and when to use each
- Choose a workflow that fits your team’s tools and constraints

BRIDGE FROM MODULE 3

What we learned yesterday:

- **What** to prompt for in Java development
- Code generation, explanation, refactoring, testing

The problem:

- ❌ Your IDE doesn’t have AI built-in
- ❌ You’re switching between browser and IDE constantly
- ❌ You don’t know when to use chat vs. inline completion
- ❌ Your team uses different tools

Today: Learn **how** to use AI tools effectively in your environment.

THE PROBLEM: MIXED ENVIRONMENTS

The reality:

- Teams use different IDEs: NetBeans, IntelliJ, VS Code, Eclipse
- Some have AI plugins, some don’t
- Some have enterprise AI, some use public tools
- ❌ No one-size-fits-all solution

The pain points:

- ❌ Copy-paste between browser and IDE is slow
- ❌ Lose context when switching tools
- ❌ Don’t know which tool to use for which task
- ❌ Inconsistent workflows across team

TWO WORKFLOWS

Sidecar: Browser + IDE (works everywhere)

Integrated: IDE plugins (requires tool support)

Both can be effective with the right approach.

SIDECAR WORKFLOW: THE NAIVE APPROACH

What happens:

1. Open ChatGPT in browser
2. Copy code from IDE
3. Paste into chat
4. Get answer
5. Copy back to IDE

Problems:

- ❌ Loses context (file names, project structure)
- ❌ Generic answers that don’t fit
- ❌ Slow context switching

SIDECAR WORKFLOW: OPTIMIZED

Keep browser and IDE side-by-side

Context header — always include:

```
I'm working in a Spring Boot 3 application, Java 17, using JPA
Project structure: standard Maven layout.
```

Benefits:

CONTEXT HEADER: PROGRESSIVE BUILDING

Level 1: Basic context (stack, version)

```
Spring Boot 3, Java 17, JPA
```

Level 2: Add file name and purpose

```
Spring Boot 3, Java 17, JPA
File: UserService.java — service layer for user management
```

Level 3: Include relevant dependencies

```
Spring Boot 3, Java 17, JPA, Spring Security
File: UserService.java — service layer
```

Level 4: Describe specific problem

```
Spring Boot 3, Java 17, JPA
File: UserService.java
Problem: Need to add validation for email format
```

SIDECAR: COPY-PASTE BEST PRACTICES

- Copy code with line numbers or file context
- Paste back with review (don’t blindly accept)
- Use diff view if possible
- Test immediately after pasting

When sidecar works well:

- ✅ No IDE plugin available
- ✅ Need to explore multiple solutions
- ✅ Working with documentation or examples
- ✅ Team uses different IDEs

INTEGRATED WORKFLOW: IDE PLUGINS

What integrated means:

- AI features built into IDE (JetBrains AI Assistant, GitHub Copilot, VS Code Copilot)
- Context-aware: IDE knows your project structure
- Inline completion: Suggestions as you type
- Chat within IDE: No browser switching

Benefits:

- ✅ Project context automatically included
- ✅ No copy-paste needed
- ✅ Diff views for applying changes
- ✅ Faster iteration cycle

INTEGRATED WORKFLOW: FEATURES

Feature 1: Inline code completion

- Shows suggestions as you type
- Accept/reject individual suggestions

Feature 2: Chat with project context

- IDE includes file structure automatically
- Can reference specific files

Feature 3: Diff views

- See proposed changes before applying
- Review line-by-line
- Apply selectively

INTEGRATED WORKFLOW: LIMITATIONS

- ⚠️ May not have access to all project files
- ⚠️ Context window limits
- ⚠️ May require enterprise license
- ⚠️ Different features across IDEs

When integrated works well:

- ✅ Team standardizes on one IDE
- ✅ Enterprise AI available
- ✅ Need fast, local edits
- ✅ Working within single file or module

CHAT VS. INLINE COMPLETION

The confusion:

- When do I use chat?
- When do I use inline completion?
- Can I use both?

Answer: Use the right tool for each task.

CHAT INTERFACES: WHEN TO USE

Use chat for:

- ✅ **Exploration**: “How do I implement X?”
- ✅ **Multi-file tasks**: “Refactor this across 3 files”
- ✅ **Design decisions**: “Should I use strategy or factory pattern?”
- ✅ **Explanation**: “What does this legacy code do?”
- ✅ **Learning**: “Explain Spring Boot dependency injection”

Chat is for exploration and design.

INLINE COMPLETION: WHEN TO USE

Use inline for:

- ✅ **Boilerplate**: Entity classes, DTOs
- ✅ **Simple edits**: Add validation, fix syntax
- ✅ **Completions**: Method signatures, imports
- ✅ **Quick fixes**: Error corrections

Inline is for localized edits.

DECISION FRAMEWORK

Use chat when:

- Exploring solutions
- Designing architecture
- Multi-file refactoring
- Explaining code
- Learning concepts

Use inline when:

- Local edits
- Boilerplate generation
- Quick completions
- Simple fixes

Use both: Chat to design, inline to implement.

MIXED ENVIRONMENTS: MAKING IT WORK

The reality:

- Team members use different IDEs
- Some have enterprise AI, some don’t
- Need consistent workflows

Strategies:

- **Shared prompt templates**: Team maintains context headers
- **Documentation**: Share effective prompts
- **Pair programming**: Cross-train on different tools
- **Policy alignment**: Ensure everyone follows data privacy rules

MIXED ENVIRONMENTS: BEST PRACTICES

- ✅ Standardize on prompt style, not tools
- ✅ Share effective prompts in team wiki
- ✅ Review code regardless of tool used
- ✅ Test AI-generated code thoroughly

Focus on consistency in approach, not tools.

SUMMARY

1. **Sidecar workflow**: Browser + IDE with context headers — works in any environment
2. **Integrated workflow**: IDE plugins with project context — faster but requires tool support
3. **Chat vs. inline**: Chat for exploration/design, inline for localized edits
4. **Mixed environments**: Standardize on prompt style, not tools

BRIDGE TO MODULE 5

What we’ve learned:

- **How** to use AI tools effectively (sidecar or integrated)
- **When** to use chat vs. inline completion

What’s next:

Module 5: RAG — connecting AI to your own knowledge base.

Tool workflows apply whether using public AI or RAG-enhanced systems.

QUESTIONS?

Module 4 — Tooling Strategies (Mixed Environment)