从模糊到确信

这个资料有点烧脑. 建议读前先深吸一口气。

前言

在工作中，我观察到自己和周围的人经常处于各种困境，具体的现象是五花八门的。我对此思考了很多，后来发现这些问题和困境都可以归结到模糊和确信这一点上。

作为 IT 从业人员，我们总是处于从需求分析到设计、编程、调试、交割的循环里，而每一个阶段都是一个从模糊到清晰的过程。我们对工作对象清晰了，心理就安定了，这是一种达到确信的心理状态。

从项目开始时的混沌，到系统交割时的井然有序，用模糊到确信来描述是非常贴切的。而把工作分割成任意小的一个单位，这种变化也一样。一天的工作是这样，一个月的工作也是。一行代码、一个 Java 类、一个功能、一个系统，都经历了同样的变化过程。学习的过程，又何尝不是如此。

然而，要实现这一点非常不容易，中间会有焦虑，有挫折，有顿悟，有释然。如何让焦虑和挫折少些，尽快达到顿悟和释然呢？这就是我写这份资料的目的。

编程是如此，世间万物其实也是。宇宙的进化、国家的变迁、人和人关系的演进莫不是如此。编程只是其中的特例罢了。如何通过洞察模糊和确信的辩证关系，来思考如何建立从模糊到清晰和确信的桥梁，建立一套有效的思维模式呢？

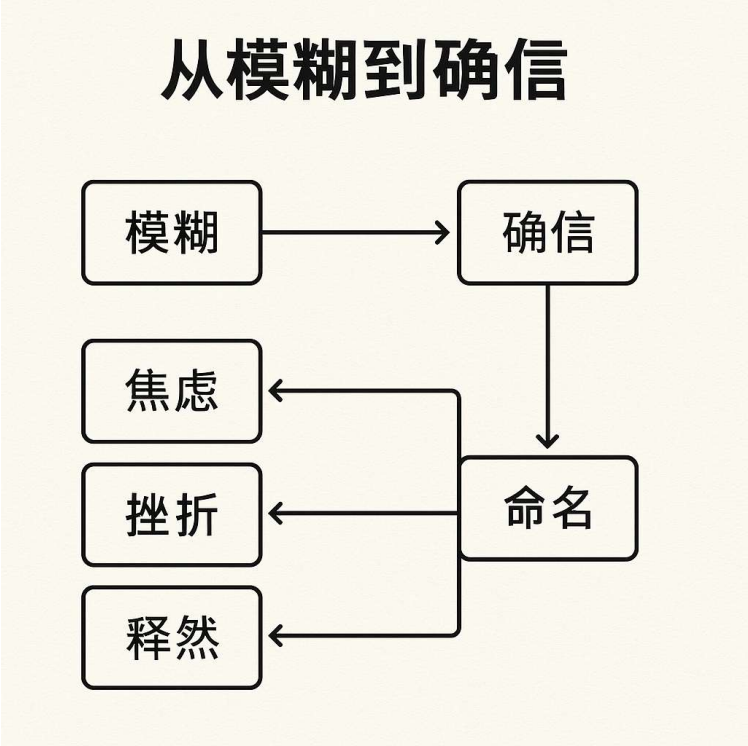
道德经的名句：“道可道，非常道；名可名，非常名。无名万物之始，有名万物之母。”其实已经为我们点明了方法的途径。不是吗？无名就是模糊的状态，名，就是给事物取一个名字。我们在编程里要起很多名字：文件名、变量名、类名、方法名、系统名、功能名、模块名；编程思想里有设计模式名、书名、算法名、数据结构名；数据库的表的名字；互联网协议的协议名；你的抖音号，也是一个名字。

起名，是思维过程一个段落的总结。而名字，是一个符号，它代表了其背后的一个概念，和一系列的认知成果。名字，也是我们在交流中不可或缺的。用“吃饭”这两个字，代替了“把食物放到嘴里以满足身体的需要……”这么一长串的文字，这是因为这两个字的意义是确定的，我们确信听者对此已有共识。难以想象如果没有名字人们该如何交流。

所以，你只要掌握了命名的能力，其实就等于掌握了思维从模糊到清晰的方法。

模糊本身就是一个名字，他促使你去反思-我是不是处于模糊状态？

确信也是。我们对从模糊到确信的各个过程都进行了命名，将试图在后面的章节里尽量把它讲明白。但是，光凭这一份资料也是万万不够的，还需要更多的细分资料来补充。这些内容，要在整个系列的资料中才能找到。即便看完后续的资料，也还是远远不够的，因为世界是如此之大。因此，希望每位读者能运用从这份资料中建立的基本方法，去探索自己领域的独特确信，这也是这个资料的意义所在。



模糊是常态但是我们需要确信

我们生活在一个不确定的世界里。

程序员、设计师、产品经理、学生、创业者，谁的日常不是充满了未知和模糊？项目目标变化、代码逻辑复杂、文档不全、需求含糊、Bug 来自天外、上司语焉不详、时间永远不够……在这样的环境里，“一切清晰、一切确定”只是幻觉，模糊，才是常态。

但我们也发现，不是每个人都会在模糊中停滞。

有些人面对模糊，先深呼吸一下，然后迅速进入状态：查信息、问问题、写笔记、画草图、验证逻辑、行动推进。他们似乎有种特殊的能力，可以把混乱的信息转化为可操作的步骤，把模棱两可的问题转化为明确的判断。

这种能力，就是从模糊走向确信的能力。

模糊不是错，但长期模糊会让你停滞

模糊本身不值得恐惧，它是探索的入口，是思考的起点，是认知裂缝中通向成长的通道。但可怕的是——我们很多人把模糊当成了终点。

“等我准备好了再做。”

“感觉差不多就行了。”

“反正大家也不确定嘛。”

“应该不会有问题吧。”

这些话，听起来平常，背后却是对模糊状态的放弃。当一个人习惯性地在模糊中绕圈，不思进、不敢问、不愿试，他就把自己的认知牢牢地绑在了低水平的漂浮状态上。

久而久之，他会发现自己做事开始越来越慢、越来越没信心，甚至开始怀疑自己的能力。

确信，是认知的“锚”

如果说模糊是一片雾气弥漫的森林，那么确信就是我们在这片森林中打下的锚点。

每一个清晰理解的点，每一个能讲明白的概念，每一个能做出决策的判断，都是认知世界中的“锚”。

有了这些锚，我们才能逐步在混乱中构建结构，在未知中生发方向感，在复杂中建立可控性。我们给每一个锚点都去取一个名字。

有了线程的概念，你就可能清楚知道某个错误是因为线程同步的问题；

你能画出一个业务流程图并给新人讲解；

你知道一个新 API 不会破坏现有逻辑，因为你理解了它的调用链；

这些“我知道”、“我确认过”、“我能说明白”，就是确信感。这种感觉不只是“我知道事实”，而是“我知道自己知道”——你有信心，有方向，有行动力。

第一章 什么是模糊，什么是确信？

我们经常说：“我有点模糊了”，“我确信这是对的”。
但——到底什么是模糊？什么又叫确信？

这两个词，听起来像是一对反义词，实际上，它们更像是一种认知的连续谱，人脑对世界的理解状态，就在这条谱上不断滑动。

一、什么是“模糊”？

模糊，不是“不知道”，而是“说不上来”。
你脑中好像有个轮廓、有个印象，但要你明确地说清楚、做判断、做决定，就开始犹豫、摇摆、含糊其辞。

✔ 简单定义：
模糊是一种对事物缺乏明确理解、判断、定位的状态。

🔍 模糊的几个典型特征：

特征	说明	示例
❓ 不确定	不知道到底是不是	“这个函数到底有没有副作用？”
🌫 模棱两可	可以这样理解，也可以那样理解	“这个‘快’是 1 秒还是 1 小时？”

特征	说明	示例
🤷 无法判断	没有标准、没有依据	“这个数据是不是正常的？”
🌀 缺乏逻辑结构	信息是有的，但你理不清关系	“这个流程怎么绕一圈又回来了？”

🔪 **一个程序员的例子：**

你在读一个 Java 项目，看到一个类 UserManager，它调用了一个方法 handle()。
你不清楚：

- 这个方法是处理登录？注册？注销？
- 是同步的还是异步的？
- 有哪些依赖？

你不是完全不知道 handle() 是什么，你只是——模糊。

二、什么是“确信”？

确信，不是“记住了”，而是“理解并能做出清楚判断”。

当你知道它是什么、为什么、怎么来的、能干什么，甚至能把它讲给别人听，那你就是在“确信”的状态中。

✔ 简单定义：
确信是一种明确、有根据、有结构的理解状态，可以做出判断、承担责任、甚至指导他人。

🌟 **确信的几个特征：**

特征	说明	示例
✅ 明确判断	“它就是这样”	“这个类的职责是管理 Session 状态”
🔗 可解释	能说明来龙去脉	“它调用了 X 模块，是为了获取权限信息”
🧠 可复述	能清晰讲给别人听	“我画张图给你说这个流程”
💡 可延伸	可以用来解决类似问题	“这个模式可以用于日志管理模块”

✒️ **还是那个程序员例子：**

你深入分析了 handle() 的实现，知道：

- 它处理的是用户注册；
- 会调用数据库、发邮件；
- 依赖配置参数、线程池；
- 错误处理策略是什么；
- 哪些地方复用它。

你现在可以写文档、改逻辑、给新人解释这个模块的行为。你处于确信状态。

三、模糊与确信的关系：一条认知之路

它们不是非黑即白，而是一个逐步推进的连续谱。

模糊（不知道、不确定）➡ 半明白 ➡ 理解 ➡ 明确 ➡ 确信

有些东西你刚接触是模糊的，慢慢通过：

- 看文档（补信息）
- 画图（建结构）
- 提问（纠偏）
- 实验（验证）
- 讲述（反向确认）

逐步转化为确信。

🧭 **总结：模糊与确信的区别**

维度	模糊	确信
感受	心里没底、说不准	有依据、有判断力
表达	说不清楚	可以讲清楚
行动	犹豫、试错、拖延	果断、计划明确
对他人	让人更迷糊	能指导别人
持久性	模糊常导致忘记	确信常能内化

📌 本章小结

- 模糊不是你的错，而是认知未完成时的正常状态；
- 确信不是天赋，而是可以通过练习获得的状态；
- 模糊是一片雾，确信是一张地图；
- 这本书，就是帮你把雾散去，把地图画清。

第二章 从模糊到确信的转换方法

“从模糊到确信”的转换，是一种认知的升级过程，也是你大脑逐步建立清晰模型、减少不确定性的过程。我们可以将它理解为一个可以训练、可以重复的流程。

🧭 一、总览图：从模糊到确信的路径

模糊（混乱、犹豫、不知所措）

⬇

看见模糊（意识到自己不懂什么）

⬇

提出好问题（定义问题的核心）

⬇

寻找信息（阅读、请教、实验）

⬇

构建模型（整理、总结、归纳）

⬇

验证模型（测试、运用、反思）

⬇

确信（明确、稳定、自信）

我们可以把这个过程称为：认知澄清六步法

🧩 二、六个阶段详解

第 1 步：看见模糊

“我好像哪里不懂，但说不清楚。”

方法：

把心里的疑问写下来（“我不知道 X 是啥”）

描述自己的不安感（“我怕出错的地方是……”）

目的：意识到“模糊存在”是第一步。

第 2 步：提出好问题

“我到底想弄清楚什么？”

方法：

把模糊的问题具体化：谁？做什么？为什么？

比如，用 5W1H 法（What，Why，How…）拆解问题

用“如果…会怎样？”来探索边界

🧠 例子：不是“我不会写 Java”，而是“我不懂 main 方法是怎么启动程序的”

第 3 步：寻找信息

“别人是怎么理解的？我可以从哪学？”

方法：

读文档、看代码、问人、查 ChatGPT

不求完美答案，先建立“初步理解”

📌注意：输入太多也会模糊，要边学边总结。

第 4 步：构建模型

“我能不能说出自己的理解？”

方法：

用自己的话讲出来（费曼技巧）

画图 / 写笔记 / 举例子 / 写类比

把碎片信息变成结构（图表、流程图）

💡 小技巧：用“为什么这样，而不是那样”来逼自己思考。

第 5 步：验证模型

“我的理解能应用吗？”

方法：

编写代码、做实验、让别人帮你验证

故意制造极端情况测试自己的理解

多角度演练（逆向推理、边界值测试）

🧪 示例：写出几种不同的 main 方法变形，测试哪些能编译、哪些不能。

第 6 步：确信

“我知道为什么这样，也知道哪里可能错。”

表现：

能清晰表达自己的判断依据

遇到挑战也不容易动摇

能教别人，也能识别错误

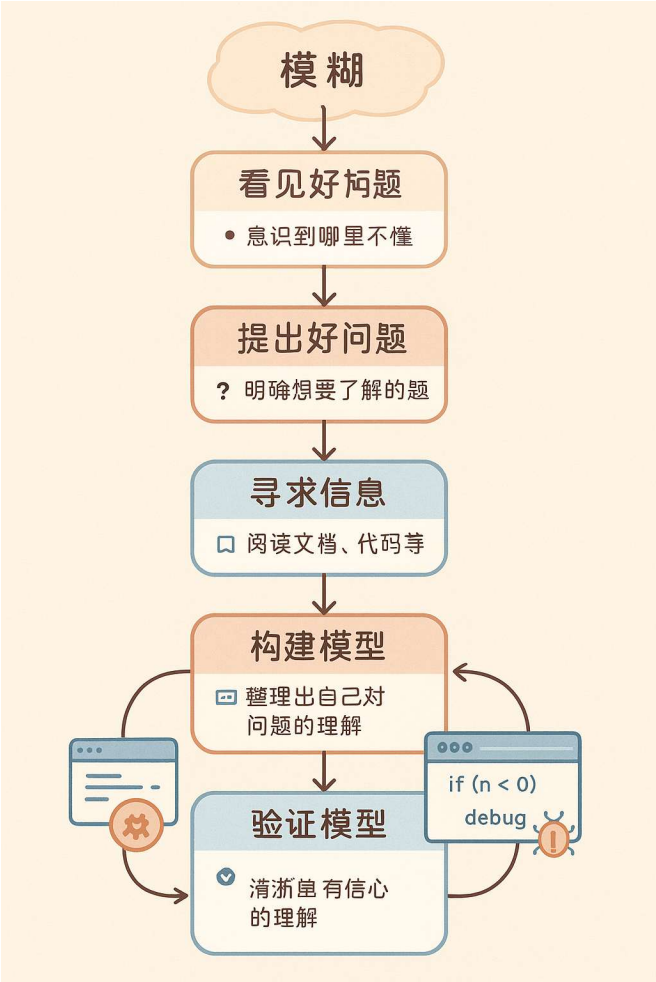
🧠 三、这是一种“技能”，不是一时的顿悟

从模糊到确信，不是一步登天的跳跃，

而是通过练习“识别模糊 → 定义问题 → 构建理解 → 验证模型”的反复训练。

它像一个肌肉训练，一次次的认知举重，

最终你能在混乱中保持清明，在复杂中看见结构。



第三章，看见模糊

“实现确信，最难的一步，是承认自己还处于模糊。”

这正击中了许多程序员、工程师、管理者乃至普通学习者长期困在认知泥潭里的核心问题——

不是不会，也不是没资源，而是心里有个声音在说：“我应该早就懂了吧。”

🧠 这其实牵涉到两个相互缠绕的陷阱：

1. ✅ 虚假确信（False Certainty）

“我大概知道吧”

“我以前见过”

“这有什么难的”

“我感觉应该差不多了”

这些表面上看起来是“信心”，但背后其实是未经验证的模糊认知假设。一旦有人追问细节、反问动机或要你画出流程图，就露馅了。

2. ✅ 情绪抵抗（Emotional Resistance）

“我不想承认我不懂”

“要是我现在问，别人会不会以为我不专业？”

“我都写了十年代码了，这东西我要是不会，多没面子”

这种抵抗让人无法说出关键的一句话：“我其实现在是模糊的。”

于是，为了维护自己“我应该懂”的形象，我们：

不查文档

不请教别人

假装明白

干脆硬写一通

最后不是崩溃，就是返工。

🔍 为什么“承认模糊”这么难？

层面　　阻力来源

认知层 不清楚自己究竟懂了多少

情绪层 面子、羞耻感、自我形象焦虑

社交层 担心暴露短板，失去信任

职业层 害怕被认为“不称职”、“不专业”

3.1 模糊的案例

◆ 找钥匙的案例

有个案例，就是找钥匙。 我女儿说，她的钥匙丢了。 我说，你好好找吧，她说，她都找过了，就是没有。 我说，要是爸爸找到了怎么办？她说，我不相信你能找到。 结果，我在她的背包里找到了钥匙。 她居然辩解说，我从来不放这里的啊。 你说气不气人？

哈哈，是的，又真实，又令人哭笑不得，而且——这正是一个绝佳的“虚假确信 + 模糊盲点 +认知自闭环”的案例，

我们一起来拆解一下：

🎭 **案例回顾：找钥匙的认知戏剧**

女儿　说钥匙找过了　自认为彻底搜索过　　虚假确信：以为找过了，其实没真正确认

父亲　提醒她再找　　保持开放态度　——

女儿　反驳：“你也找不到”　认知封闭（认定结果）　认知闭环：错认为“没找到=找过了=没有”

父亲　在背包中找到钥匙　　用实际动作验证假设　——

女儿　辩解：“我从不放这里的”　　为了保护面子自我合理化　　认知防御机制

🧠 **这个案例中暴露的认知问题**

1. 虚假确信：

她以为自己已经彻底找过了，错把“有印象”当作“验证过”。

关键词：“我确定我看过了”、“我记得不会在这”、“我一早就找了”

➡ 实际上都没动手查证，只是依赖记忆或印象。

2. 认知盲区：

她从未考虑到“也许自己会临时改变习惯、放错位置”的可能性。

对自己的行为也产生模糊。

3. 自我封闭认知闭环：

她先得出结论：“钥匙不在”，再用“我找过了”来反证这个结论，形成封闭循环。

没找到 → 所以不在 → 所以不用找了 → 你找不到 → 我没错

→ 逻辑自闭

4. 认知防御机制：

当你找到了，她说“我从不放这里”，这是一种认知（不协调）处理机制，为了维护自己的“我没错”形象。

🔍 如果这是程序员，结果可能是这样的：

“我看了代码，肯定不是这里出的问题”

→ 结果你加个日志一试，就发现出问题的正是这里

→ 他回一句：“这地方以前从来没出过问题啊……”

是不是感觉似曾相识？😉

♦️ 案例 2

还有人说，我觉得什么什么，其实他的那个觉得不知道从哪里来的，也许是直觉。

比如，做过 SWIFT 系统的人都知道，Swift 电文有局部的雷同，比如 MT300,MT202，都有 Tag56.

结果，他把应该在 MT202 的操作，在 MT300 上也实施了一次，而且事先没有请示。

这又是什么模糊呢？

这是一个非常典型、危险程度极高的“类推型模糊”（analogical fuzziness），它同时伴随着未经验证的直觉操作与过度迁移认知。我们可以将它归入“从模糊到确信”路线上的一种高级模糊误区。

🧠 这个案例的核心行为：

因为 MT300 和 MT202 都有 Tag 56，于是“我觉得”它们可以做相同的处理，

结果把 MT202 的操作逻辑，未经验证地“套用”到 MT300 上，

并且没有请示或二次确认。

👄 模糊类型判定：类推型模糊 + 直觉确信 + 缺失验证

模糊类型	表现	危害
✅ 类推型模糊	拿“相似结构”当作“相同行为”	系统可能行为完全不同，误操作严重

模糊类型	表现	危害
✅ 直觉式确信	“我觉得应该可以”但无法解释为什么	不可重复、不可验证、不负责任
✅ 缺失验证	没有查文档、没有测试、没有请示	错误没有在行为之前被拦截

类推型模糊的结构特征

类推成立条件（认知科学角度）	错误类推的偏差
有共同结构 + 有共同语义 + 有相似用途	仅看到表面字段雷同（Tag 56），忽视上下文与业务目的
迁移逻辑可以验证	直接跳过验证，靠“经验”或“感觉”做判断
存在明确差异边界	没有意识到 MT202 是清算指令，MT300 是衍生品交易确认

♦ 案例 3

还有一种，经验的复用问题。 我的一个部下，他在别的现场用了一个操作，上传了多个文件。 结果，他用同样的方法在我的工作现场未经许可也做了同样的操作，结果产生了不良的后果。这种要怎么防范？

这个案例，非常具有普遍性。它揭示了“经验的无意识复用”在工作中的风险。这类错误看似是“有经验”的表现，实则是**经验没有经过语境校验就盲目迁移的认知偏差**。

这个问题的本质：经验迁移失控

可以归类为：

🔁 **模糊类型 ：未经语境验证的经验迁移**

🧠 **案例解析**

场景	行为	后果
A 项目	部下上传多个文件，系统正常处理	一切运行良好（经验形成）
B 项目	他在**未经请示**的情况下复用了相同操作	触发了错误流程、数据污染或权限越界

🔬 **为什么这不是“有经验”，而是“模糊”？**

问题类型	描述
语境模糊	他未理解不同项目环境中的系统设定、流程、规则是否一致
边界模糊	他不知道哪些行为在此现场是“可接受的”，哪些是“需要许可的”
验证缺失	他没有确认“上传多个文件是否被允许”，是否有白名单、路径规则、版本要求等

✅ 他不是“经验主义者”，他是**经验泛化者**，而经验**必须在新语境中被验证**，才可被安全迁移。

⚠️ **为什么这是一个危险的模糊点？**

- 表面上看是“积极主动”，其实是“未经许可的越权行为”
- 他自己**以为做对了**，事后还可能为自己辩护：“以前我就是这么做的”
- 这不是行为问题，而是**认知模型未更新**

🛠 如何防范这类经验迁移型模糊？

一、在组织层面建立“语境再验证机制”

对象	防范策略
有经验的员工	教育他们：经验不能直接迁移，必须经过场景确认、制度确认
所有员工	对跨项目操作设立显性“许可边界”：“此类操作必须经过批准”
项目文档	明确写清楚：哪些行为属于“允许经验迁移”，哪些需重新评估

二、制定经验迁移的“五道确认”

五问经验迁移检核法：

1. **原场景和当前场景有哪些不同？（系统、版本、权限、目标）**
2. **原操作是否有上下文依赖？（前提条件是否一致）**
3. **当前场景是否有明确许可？**
4. **是否做过小范围测试？**
5. **是否向项目负责人/团队通报并备案？**

只有五问都满足，经验迁移才是安全的。

🔷 案例 4

同事说：“我记得上次我也是这么操作的。”

我看了一下部署日志，那次环境是测试版、版本是 v1.1，配置有个 flag 默认启用；
而这次是生产环境、v2.0，flag 默认关闭，当然不一样了。

他不是乱来，他只是陷入了记忆错觉。

他记得某个片段，却忘了关键条件；
他记得结果，却忘了环境；
他以为自己“记得”，其实只是脑中残存了一张“不完整的地图”。

这个案例精准地指出了**另一个非常隐蔽却非常普遍的认知模糊类型**，这次我们可以称之为：

🧠 「记忆错觉模糊」（Memory Illusion Fuzziness）

又名：**“片段记忆导致的错误复现”**

✅ 例子结构如下：

- 同事说：“我记得上次是这么操作的，应该是对的。”
- 实际上：他**只记住了部分步骤或环境条件**，而这次的情况已不同。
- 结果：同样操作，却得到不同结果，预期落空。

✨ 这类模糊的核心问题不是“他不知道”，

而是——**他以为他记得很清楚，但其实他忘了关键部分，或者把两次情境混淆了。**

📌 **这种模糊的三大成分：**

模糊维度	表现
记忆片段化	只记住“关键动作”，忘记隐藏的依赖条件（如环境变量、配置项）
记忆混淆	把两次不同的事件混在一起——“上次在 A 系统，这次在 B 系统”，但脑子里合并了
回忆带入错误信心	因为“我以前成功过”，对当下操作产生虚假确信感

🧠 **常见表现：**

- “我记得就是这样部署的啊。” ➡ 忘了那次是 debug 模式
- “我上次也是这么测的。” ➡ 忘了那次数据库没加权限控制
- “之前接口也是这么调用的。” ➡ 忘了那次 token 没过期
- “我记得路径写 /data/ 就可以了。” ➡ 忘了那次加了软链接

⚠️ **本质上，这是一种「回忆驱动型确信偏差」：**

原因，后果

记忆感觉真实 ➡ 信以为真 ➡ 跳过验证步骤

成功经验错记 ➡ 新任务照搬 ➡ 失败而不自知

环境变了 ➡ 操作没变 ➡ 得出“为什么这次不行了”的困惑

✅ **如何防范“记忆错觉”模糊？**

方法	说明
记录真实操作	代替“我记得” → “我查看了操作日志 / README”
写下每次实验条件	包括版本号、参数、路径、配置状态
养成验证流程	即使“记得”，也要再次确认是否适用于当前环境
避免自问式决策	不说“我记得”，而是说“这一次我确认了什么”
版本化一切	包括脚本、文档、数据样例等

记得”是一种幻觉，
它最容易骗过你自己。
清晰不是回忆来的，是重建和验证来的。

上述案例的共同点是，都没有意识到自己处于模糊状态，这是一种元认知的缺失。

毛泽东说过，人贵自知之明，所谓自知，就是一种自己对自己做出正确评价的能力。

这不是一句空话，它揭示了“认知的自我校准能力”的极高价值。

真正有智慧的人，不是“总是对”，而是“能知道自己哪一点可能是错的”。
不被经验所困，不被情绪所挟，不被身份所绑。

自知的第一个动作，是敢于说：“我现在可能不懂。”

这句话的心理代价极高——
它要求你撕掉“专业”“资深”“可靠”的面具，
让你以“重新开始者”的姿态面对工作、判断、系统与人群。

但这一步，恰恰是从模糊走向确信的唯一入口。

中国古代有很多寓言，刻舟求剑，掩耳盗铃，守株待兔，都是对处于模糊而不自知的一种讽刺，我们要时刻清醒，不要掉入这种危险的认知陷阱。

我相信，只要大家能够时刻保持自己有可能出错的清醒，那么就离成长不远了，剩下的就是只要正确的运用后述的方法就可以了。

第四章，如何问问题——把模糊压缩成清晰的语言

✅ 为什么要从“会问问题”开始？

因为：

- 你问得越模糊，查到的信息就越泛。
- 你问得越清晰，别人越能帮你，你自己也越容易找到答案。
- 问题越具体，你的认知模型也越成熟。

一句话：

“不会提问的人，就是不知道自己到底哪儿不懂。”

🧰 程序员的提问四步法：SPQR 模型

步骤	名称	含义	示例
S	Situation（情境）	当前的问题发生在哪儿？	“我在 Spring Boot 项目中热更新配置”
P	Problem（现象）	遇到什么具体的异常？	“改了 application.yml 但配置没生效”

步骤	名称	含义	示例
Q	Question（提问）	你想搞清楚什么？	“为什么配置不自动 reload？有没有触发机制？”
R	Research（尝试）	你查过/试过了什么？	“试过 actuator/refresh，试过 config reload 但无效”

—

问问题前先把这四点写出来，你的提问就会从：

❌「为什么配置改了没用？」

变成：

✅「我在 Spring Boot 项目中使用 application.yml 配置了 xxx。改动后发现配置未生效。我试过 actuator 的 refresh，但没有触发更新。请问这种情况下还有哪些触发 reload 的方式？」

—

这已经是从“模糊语言”走向“结构化思维”的开始。

🧠 典型的“坏问题” vs “好问题”对比

坏问题	好问题
“程序出错了，怎么办？”	“运行 A 方法时抛出 NullPointerException，调用栈如下，我不确定哪个变量为 null。”
“这个函数不能用了”	“调用 addUser() 返回 false，查看日志发现数据库连接失败，排查到配置项 DB_URL 是否正确？”

坏问题	好问题
“我写了个接口访问不了”	“我在本地部署的 Spring Boot 接口 GET /user 返回 404，但 Controller 已加 @RequestMapping。路径是否区分大小写？”

—

每一次“说不清”的地方，都是你还处于模糊的地方。

—

🔧 提问辅助工具：模糊点定位表

我现在的困惑点在……	自查问题示例
输入是怎么给的？	参数格式对了吗？Header 设置正确吗？
系统现在是什么状态？	是否有缓存？是否部署的是最新代码？
我的预期是？实际是？	是不是期望错了？还是行为变化了？
我尝试过什么？结果是什么？	日志有信息吗？我能复现吗？

—

建议读者养成“写提问草稿”的习惯，把“脑子里的模糊”写成“纸面上的问题结构”。

📘 段落建议

- “清晰提问，是认知清晰的第一试金石”
- “语言是压缩模糊的工具”
- “先说清楚你哪里不清楚”

✅ 你问问题的质量，决定你能走多远

- 会问问题，代表你能整理混乱；
- 能问出好问题，代表你已经接近答案；
- 问得越来越具体，代表你认知越来越清晰。

🧠**「不是问一个问题，而是问一个系列的问题」**

提问本身是从模糊到清晰的过程

✅ **为什么“第一个问题”往往是模糊的？**

1. 因为你的认知还在混沌中，只能靠现有词汇试着表达；
2. 因为你还没分清“现象、原因、期望、边界”；
3. 因为你其实还不知道自己真正想问的是什么。

—

📜 比如：

你第一个问题是：「为什么我的服务启动不了？」
看似合理，但实际上它模糊点包括：

- 什么服务？
- 启动时发生了什么现象？
- 日志提示什么？
- 你是否做了前置操作？
- 你期望它表现如何？

—

这时候你要做的，不是“急着等回答”，而是自己反问自己：

- 我的问题具体吗？

- 我描述完整了吗？
- 我问的是现象，还是原因？
- 有没有可能我问错了问题？

🎯 提问的五级递进模型（从模糊到确信）

等级	提问类型	特征	示例
1	模糊问题	范围大、语焉不详	“为什么不行？”
2	现象式问题	指出具体行为或症状	“启动时报错，log 中提示端口被占用”
3	假设式问题	带有分析假设	“是不是因为系统默认端口没释放？”
4	边界式问题	明确限定条件和场景	“在 Windows 下，使用 Jetty 启动时占用 8080 端口是否会导致...”
5	验证式问题	可测试、可复现、可验证	“是否通过更改配置将端口改为随机后可成功启动？”

—

第一级是模糊，
第五级是确信，
你每往下问一步，认知就清晰一层。

🧰 使用「反复提问法」：

每当你问出一个问题，就自问下一句：

1. “我这个问题问清楚了吗？”
2. “我漏了哪些上下文？”
3. “有没有更小、更具体的子问题？”
4. “这个问题能被验证吗？可以复现吗？”
5. “我是在问‘症状’，还是在问‘根因’？”

—

这其实也是一种“认知调试”。

📘 **总结：**

很多人以为，“问题问出来，就该得到答案了”。
但事实是：真正的问题，往往要反复追问，才能从“模糊感”里走出来。

第一问，只是点亮黑暗的一支火柴；
第二问，才能看清桌子的轮廓；
第三问，才敢开始走向那个门口。

不断提问，是从混沌中打磨逻辑边界，是你和世界“协作建模”的过程。

—

✨ **建议：**

问题不是你扔出去的飞镖，而是你铺设的阶梯。
每往下问一步，世界就清楚一分。

第五章，如何查找信息：用问题引导搜索

✅ 一、查找信息的最大误区是什么？

很多人即便提了好问题，也查不到好答案，原因是：

错误行为	为什么低效？
用“自然语言”去 Google / ChatGPT	模糊关键词，系统无法定位核心
把错误现象直接复制去搜索	可能查到的是别人的问题，不是你的上下文
想直接得到“最终解答”	查找是构建认知，而非搬运答案
不看文档、只搜博客、抖音、小红书	失去对原始语义的掌控，只得到“转述后的碎片”

—

📌 查找不是“搜索答案”，而是“搭建认知模型的过程”。

🧠 二、查找信息的四步法：R-A-T-E 框架

步骤	含义	行动
R - Refine 重新聚焦关键词	从“问题描述”中提取核心词	e.g. Spring + YAML + reload + actuator
A - Ask 明确自己要找什么	是找配置说明？使用样例？报错分析？	写出问题类型
T - Target 针对平台 / 文档	对应使用：官方文档 / Stack Overflow / GitHub issue / ChatGPT / MDN / etc.	site:spring.io + “reload”

步骤	含义	行动
E - Evaluate 验证信息可信度与适用性	是否是最近 2 年？是否适用于你的版本？是否可以复现？	检查环境/版本/语言等是否一致

—

光查到答案没用，要能判断“它对不对”“适不适合”“为什么对”。

📘 三、查信息的几个常用策略

情境	查找策略	工具建议
查概念/用法	关键词 + “官方文档”	site:docs.python.org, MDN, man7.org, etc.
查报错信息	错误关键词 + 项目名 + 框架名 + 版本	Google + StackOverflow
查配置参数	项目名 + 参数名	site:spring.io + “server.port”
查语义差异	“xxx vs yyy”	“List vs Set in Java”
查类似经验	GitHub issue / ChatGPT / 博客 + 自己版本环境	“redis timeout spring boot site:stackoverflow.com”

—

🎯 教用户一招：加 site:xxx.com 来限制搜索域，可以避免泛滥的博客和内容农场。

✨ 四、查到信息后，别急着用——要验证！

任何你找到的内容，必须经过验证：

1. 它适用于哪个版本？我的环境一致吗？
2. 它有没有隐藏副作用？
3. 它的假设条件是什么？我满足了吗？
4. 有没有“必要信息我还没查到”？（例如配置顺序、依赖注解）

—

✅ 这一步可以结合“构建最小实验环境（Minimal Reproducible Environment）”进行验证。请看后续章节。

第六章，从查到答案，到验证理解：让知识变成确信的过程

你已经把认知推进到关键的最后一公里：

查到答案，并不等于你已经“理解”或“解决了问题”。
真正的“确信”来自于：✅ 自己验证过、确认它在你这个具体情境下真的有效。

一、为什么一定要验证？

因为信息本身可能存在偏差或不适用：

信息来源 潜在问题

ChatGPT 提供的是大概率解释，但不一定与你的上下文吻合

搜索结果 可能适用于旧版本、不同环境或误解上下文

信息来源 潜在问题

博客文章 可能含有作者未觉察的偏见或约束条件

官方文档 有时语焉不详，或需要组合多个点理解其行为

他人经验 适用于他，但不适用于你

—

🧠 所以你必须训练自己：

不接受信息，只接受验证过的理解。

二、验证的核心目标

查到一个信息后，你要回答这三个问题：

1. 它在我的当前环境中是有效的吗？
2. 它的行为与我预期是否一致？有没有副作用？
3. 我是否理解它为什么能/不能起作用？

—

📌 一句话总结：

✅ 可运行

✅ 可解释

✅ 可迁移

这才是“知识已确信”。

三、验证的五步法：V-A-L-I-D 模型

步骤	含义	操作
V - Verify 环境一致	验证版本、语言、框架、系统一致性	比如 Python 3.11 vs 3.6 的差异，Spring Boot 2 vs 3
A - Apply 构建最小测试案例	在沙箱环境中重现解决方案	搭个临时项目或 test class 试试
L - Log 比对行为与期望	打日志、输出中间状态、观察差异	是否按预期触发？副作用是否发生？
I - Isolate 分离变量	每次只验证一个变量/配置项	避免“改了一堆不知道哪个起作用”
D - Document 写下验证结果	总结现象 → 推理机制 → 成果笔记	“这个参数必须在 xxx 之后设置，否则不生效”

—

一个心法：

不验证的“答案”，只是别人的假设。
经验证过的知识，才是你的理解。

四、如何验证：常见类型与方法

情境	验证方式
参数配置是否生效	修改配置项 → 重启服务 → 看日志/行为变化
函数调用是否如预期	打断点、打印入参出参，查看路径
页面是否响应正确	用 Postman / curl / devtools 发请求观察响应体
权限是否控制生效	用不同用户角色登录尝试访问
异常是否被 catch	制造异常场景，看日志输出是否符合预期

情境	验证方式
新逻辑是否覆盖旧逻辑	结合单测或覆盖率工具分析路径是否走通

—

🎯 如果你要用 AI 给出的建议，也必须自己搭建“最小验证环境”。

验证环境是指，就算错了，也不会造成损失的环境。这一点非常重要。

比如，你的实验有可能删除数据，或文档，那一定要保证这些动作不会造成破坏性结果。要做好备份，保证在最坏的结果之下也能恢复数据或这文档。

📘 **小节标题建议：**

- 「查信息只是开始，验证才是结束」
- 「让知识落地的最后一步：亲手确认」
- 「记住：没有验证，理解就是幻觉」

✨ **推荐：**

看别人说“可以”不算数，
你要能说：“我试过了，确实可以，而且我知道为什么。”

第七章，认知澄清不是直线，而是循环

问题、信息、验证的三元螺旋结构

✅ **一、线性认知的幻觉：很多人以为是这样的**

看似流程：

1. 提一个问题
2. 去查信息
3. 得到答案
4. 应用 → 完结！

❌ 这是假象。这种“输入-输出式”的线性思维适用于标准化问题，但在现实开发、设计、调试、学习中，几乎都失效。

—

现实更像是这样：

第一个问题 → 查了 → 得到一点信息 →
但发现理解还不够 → 于是回头改问题本身
→ 再查 → 再验证 → 结果又失败 → 再修正问题表述 → … → 最终靠积累构建确信。

这才是真实的认知过程。

—

🧠 二、认知的三元螺旋模型：P-Q-V 循环

你可以给读者介绍这个模型：

步骤	含义	行为
P - Problem	提问，定义模糊处	澄清你到底不懂的是什么
Q - Query	查找信息，构建解释	利用文档/AI/搜索定位候选答案
V - Validate	实验验证，观察行为	比对预期和实际，确认认知准确性

这个循环并不是一圈走完就完事了，而是：

- 每走一圈，你的理解就更清晰一点；
- 每次失败，都会逼你“回到 P”，重新定义问题；

- 最终，你不是靠“一次查到对答案”，而是靠“多轮循环，逼近真相”。

—

这就像钻螺丝：

每转一圈，不是在原地，而是越来越深入。

—

📘 **示例说明：**

🍼 举个例子：

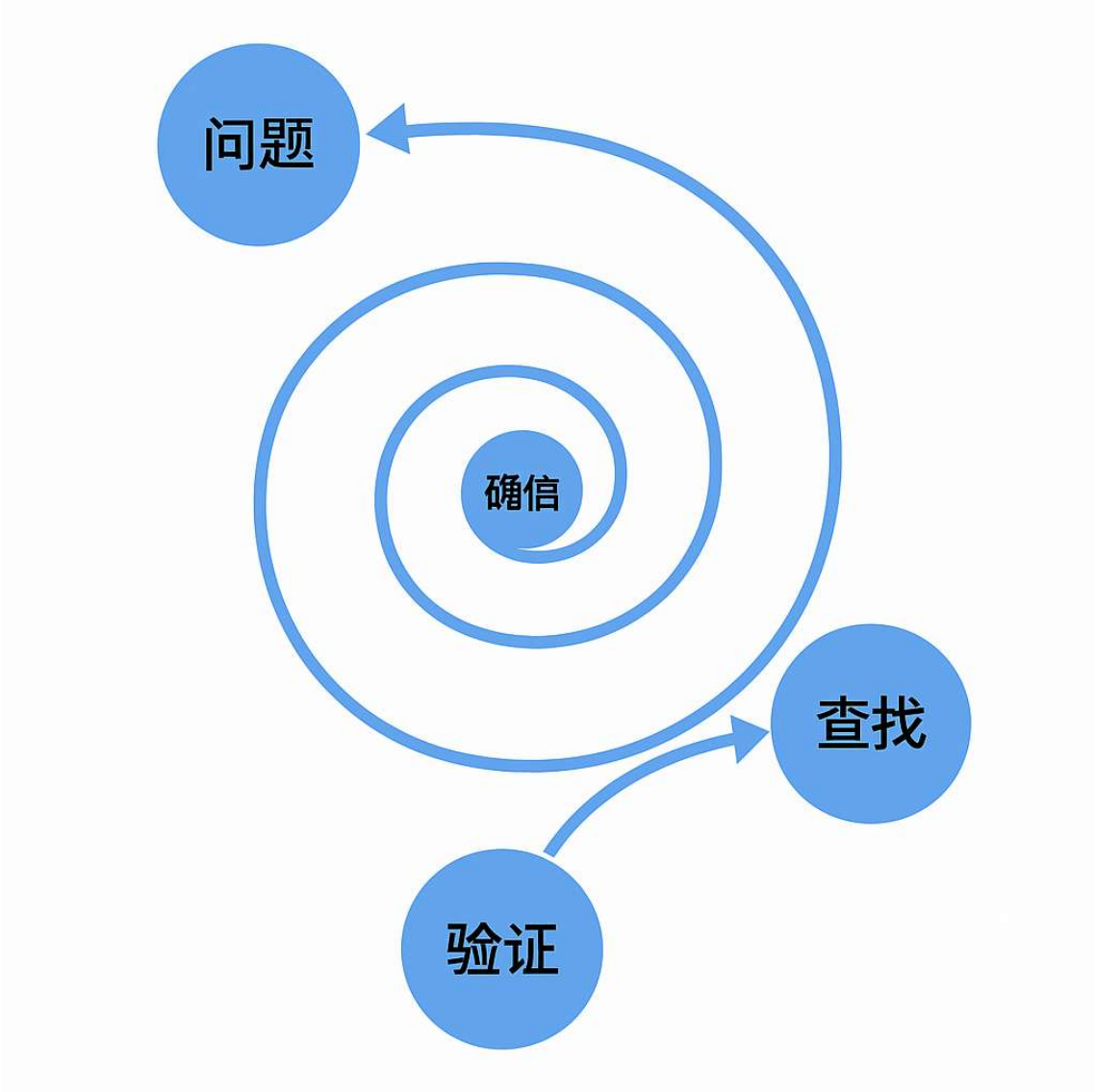
1. 你问：“为什么系统登录后跳转失败？”
2. 查资料说是 session 丢失，建议检查 cookie 设置；
3. 你试了，但发现设置无效；
4. 于是你意识到：原问题问得太粗，应该是“为什么 sessionId 不保持？”
5. 再查时加上了 nginx、跨域、set-cookie 等关键词；
6. 最终发现是代理配置 strip path 引起的路径错乱；
7. 到这时，你对整个 cookie、session、代理行为，有了系统认知。

—

这一整套过程，是三个环节的反复迭代，而不是“查一下就结束”。

—

🧭 **图示建议：认知循环图（我可以为你绘制）**



✨ **段落总结：**

知识不能被一次问出来，只能被反复打磨出来；
确信不是答案，是一个不断验证后剩下的核心。

当你愿意回到原点重问问题，你就已经在通往清晰的路上了。

第八章，警惕过度确信

——当你觉得懂了，其实只是另一个起点

我们已经用放大镜，逐步解构了“从模糊到确信”的认知旅程。
你也许开始意识到，模糊不是你的错，确信也不是一蹴而就。

但请注意一个悖论：

你此刻对“从模糊到确信”这件事的理解，本身仍然是模糊的。
读完这个资料，不等于你掌握了它；
它只是你走上这条路的起点。

你现在对这些概念的理解，也许还是碎片的、含混的，甚至可能比没读之前更迷茫——这是正常的。
因为你刚刚经历了“认知扰动”阶段：老的模型被打碎，新的结构还没建立。

所以：

✅ 请不要急着“看懂全部”
✅ 而是挑出你“稍微明白一点点”的部分，去试试看
✅ 然后回来，再读一次，再理解一次，再修改你的认知地图

你不需要完全懂，只需要开始“使用它”。
在实践中，你会一点点，从模糊中长出自己的确信。

—

当有一天，你发现自己不再畏惧模糊，能稳稳地从模糊中抓出问题、搭出模型、构建解法——那就是你真的“学会了”这本书。

那时，你的编程能力、问题处理能力、团队交流能力，也一定已经跃升到一个新层次。

但请记住：这不是终点。

🌀 所有你此刻的确信，也都将是未来新的模糊的起点。
🧠 不断拆除旧认知、建立新理解，是我们作为“清醒的程序员”的生命姿态。

—

所以：

- 请反复使用这本书的框架。
- 请把这份思考，传递给你身边的同事与后辈。
- 请警惕“我已经知道了”的念头。

因为：

“真正危险的不是模糊，而是你以为你已经看清了。”
“与时俱进，不是选项，而是生存方式。”

祝你好运！

写在后面

有本书叫做，拆掉思维里的墙。

《拆掉思维里的墙》，是一本在中国颇具影响力的认知类畅销书。它由作者古典撰写，强调的是：

人的很多问题，来源于“自己给自己建起的思维限制”，
而“拆墙”则是一种打破固有认知、重新定义自我与世界关系的过程。

《从模糊到确信》之间的共鸣点：

《拆掉思维里的墙》　　**《从模糊到确信》**

聚焦的是**心理认知壁垒**　　聚焦的是**程序员的认知模糊**

《拆掉思维里的墙》　　**《从模糊到确信》**

“墙”是看不见的思维限制　“模糊”是看不清的问题边界

主张**反思原有信念**　　主张**识别虚假确信和错觉**

典型口吻是心理启发式　　典型口吻是认知系统工程化

两者都是在帮助人类做一件事：

从“被自己的脑骗住”中走出来。

✨**《拆掉思维里的墙》连接点：**

“人不是输在不会，而是输在**不愿重新理解**；
模糊之下，是思维之墙，
而确信，则是拆墙后的视野。”

✅ 警示：

- 过度确信不只是阻止你获取信息，而且**阻止你质疑信息的来源**；
- 它不是让你变笨，而是让你变得“只相信自己旧版本的聪明”；
- 它不是黑暗，而是“你一直以为的光明”，所以你从不怀疑它。**“过度确信，是思维之墙最坚固的砖”**
- **“从信以为真，到重新认知”**
- **“经验不是高墙，是应该拆解重构的脚手架”**