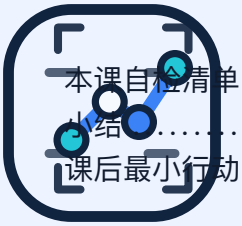
Week02｜课时 3｜多模态最小元数据与 PII 分级：文档、音频、视频、工单怎么统一

Table of contents



metadata 不是备注，而是后续检索、引用、权限和审计的 runtime interface

这一讲先不急着把所有规则压进 contract，而是先把更底层的一层统一起来：

不同模态进入系统时，最低要带什么上下文，哪些字段必须带着 PII 动作一起进入后续门禁。

如果这一层没有先统一，后面的 Data Contract、Manifest、采集门禁、RAG 引用、权限过滤和审计追责，都会被迫回补。

回看课时 2 进入课时 4 返回 Week02 总览

下载讲义

提供适合离线阅读的 PDF 版和适合批注整理的 Word 版。

PDF 版 · 打印 / 离线阅读 Word 版 · 批注 / 二次整理

这节课解决什么问题

到这里，你已经完成了 docs/blueprints/week02/asset_inventory_v1.csv 的首版盘点。

但这还不够。

因为 AI 系统真正要消费的，不是“这家公司有 PDF、有工单、有录音、有视频”这样的目录信息，而是：

- 这条内容**来自哪里**
- 它能不能被**回指到页码 / 时间戳 / 章节 / 话轮 / 关键帧**
- 它能不能被**检索、送进模型、展示给用户、驱动工具动作**
- 它里面哪些信息必须**脱敏、降权、隔离或直接拦截**

所以这节课的任务，是把上一讲的输入地图，继续压成两套真正能进入工程系统的基础规则：

1. **多模态最小元数据标准**
2. **字段级 PII 分级与动作矩阵**

参考学习时间（60–75 分钟）

如果你只阅读正文，大约需要 35–45 分钟；如果你跟着本课一起把四类输入的最小 metadata、PII 动作矩阵和样例记录真正收进 repo，建议预留 60–75 分钟。

本课你将完成什么

学完这一讲，你应该能做到：

1. 解释为什么 page_no / bbox / section_path / start_ts / speaker_role / frame_ts 这些字段不是“可选增强”，而是证据链基础设施。
2. 用一套统一方法定义 ticket / document / audio / video 四类资产的最小元数据。
3. 把 PII 从“有或没有”升级成“字段级分级 + 系统动作”。
4. 在仓库里写出首版 docs/blueprints/week02/metadata_minimums_v1.md 和 docs/blueprints/week02/pii_policy_matrix_v1.csv。
5. 准备一份 tests/contract/fixtures/week02/sample_records.json，为下一讲 JSON contract 和 contract tests 做样例输入。

本课产出

完成这一讲后，你至少应该产出这 3 个工件：

- docs/blueprints/week02/metadata_minimums_v1.md
- docs/blueprints/week02/pii_policy_matrix_v1.csv
- tests/contract/fixtures/week02/sample_records.json

先看一张总图

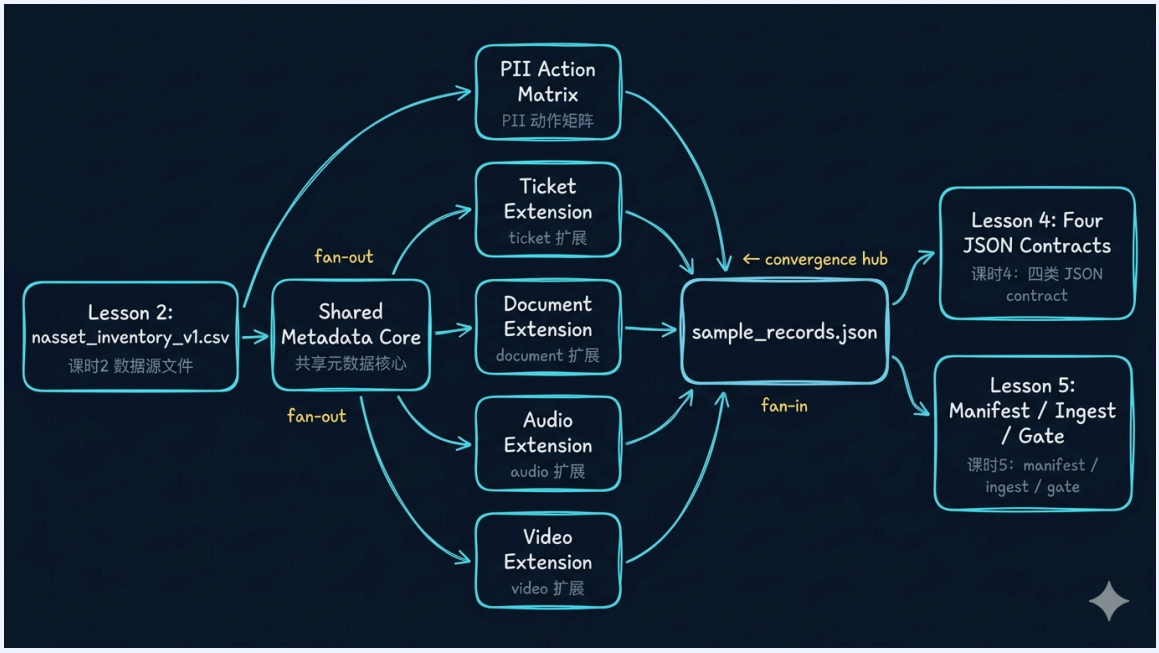
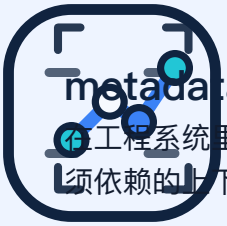


Figure 1: Week02 课时 3 总图



metadata 到底在运行时被谁消费

在工程系统里，metadata 不是为了“写得更完整”，而是为了让后续不同模块都拿到自己必须依赖的上下文。

metadata 字段	谁会消费它	影响什么
access_scope	retrieval / tool layer	权限过滤
page_no / bbox / section_path	citation / audit	引用可回指
speaker_role / start_ts / end_ts	transcript QA / HITL / review	对话责任、片段定位
schema_version	contract / compatibility check	演进感知
pii_level	policy engine	是否可入模、是否可展示

为什么 metadata 会直接决定检索过滤

主流检索系统已经不是“把正文全塞进索引”那么简单，而是：

- 根据 access_scope / tenant / role / product_line 过滤
- 根据 asset_type / doc_type / source_system 缩小范围
- 根据 pii_level / license_tag 决定哪些结果根本不能参与召回

如果这些字段没有进入 runtime metadata，系统就只能靠“先召回再补救”，这通常已经太晚。

没有的字段	直接影响什么
access_scope	过滤失效，可能越权召回
tenant_id	多租户边界失效
product_line	错产品线的文档和工单被混到一起
license_tag	本不允许展示的内容被拿来回答

为什么 metadata 会直接决定引用与追责

很多系统“看起来能答”，但不具备可追责能力，问题通常不在模型，而在证据字段。

证据字段	它解决什么问题
doc_version	你引用的是哪个版本

证据字段	它解决什么问题
page_no	你能不能回到正确页
section_path	你能不能定位到正确章节
bbox	你能不能把答案对到正确区域
speaker_role	你能不能分清客户说了什么、客服说了什么
start_ts / end_ts	你能不能回到音频/视频的正确时间片段

如果这些字段缺失，系统就会出现一种很危险的状态：

回答听起来像对的，但没人能证明它到底来自哪里。

真正的统一方法：共享核心 + 模态扩展

shared core + modality extension

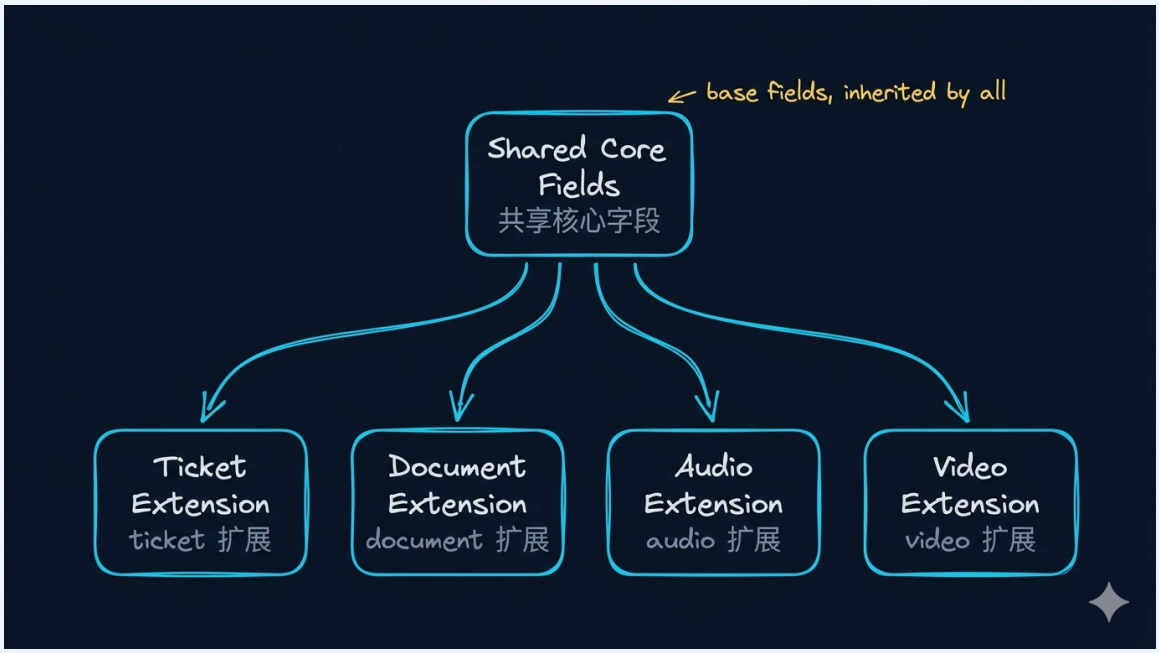
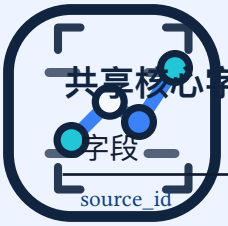


Figure 2: Week02 共享核心 + 模态扩展图

为什么最小 metadata 必须先于 chunking

- 没有最小 metadata，切块再精致也无法稳定回指。
- 没有 section_path / page_no / start_ts / frame_ts，后面只能“像是从这里来的”。
- 这会直接影响 Week07 的证据链和 Week08 的 RAG citation。

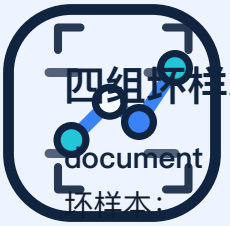


共享核心字段

字段	你为什么需要它
source_id	标识这条记录属于哪类输入源 / 契约对象
asset_type	告诉系统它是 ticket / document / audio / video 哪一类
source_system	知道它来自哪个业务系统或内容系统
source_fingerprint	用于判重、回指、版本对齐和审计
schema_version	让系统感知兼容性与演进
owner	校验失败时先找谁
access_scope	谁能看、谁能搜、谁能送进模型
pii_level	合规动作入口，不是备注
observed_at	这条记录对应的业务时间、版本时间或观测时间

扩展版四类输入最小 metadata 清单

资产类型	共享核心	模态必需字段	如果缺了会怎样
ticket	source_id / asset_type / source_system / source_fingerprint / schema_version / owner / access_scope / pii_level / observed_at	ticket_id / tenant_id / status / opened_at / updated_at / requester_role / product_line	状态语义和增量判断容易漂
document	同上	doc_id / doc_version / page_no / bbox / section_path / license_tag	回指不稳、citation 失真
audio	同上	call_id / speaker_role / start_ts / end_ts / confidence / pii_redaction_flag / transcript_text	对话责任和脱敏状态不明确
video	同上	video_id / segment_ts / frame_ts / transcript_ref / image_caption / ocr_text	视频时序和视觉证据丢失



四组坏样本 vs 好样本 JSON 对照

document

坏样本：

```
{
  "asset_type": "document",
  "doc_id": "DOC-337",
  "text": "升级失败后请重启设备。"
}
```

好样本：

```
{
  "source_id": "knowledge_doc",
  "asset_type": "document",
  "source_system": "kb_portal",
  "source_fingerprint": "sha256:doc-337",
  "schema_version": "v1",
  "owner": "product-ops",
  "access_scope": "product_line",
  "pii_level": "public",
  "observed_at": "2026-04-14T09:05:00Z",
  "doc_id": "DOC-337",
  "doc_version": "v3.2",
  "page_no": 37,
  "bbox": [72, 128, 488, 220],
  "section_path": "升级/失败恢复",
  "license_tag": "course-safe"
}
```

为什么坏： – 没有 page_no / bbox / section_path，后续无法稳定引用。 – 没有 license_tag，无法判断展示和分发边界。

后果是什么： – citation 会失真 – entitlement 会失控 – 审计时很难回指原文位置

audio

坏样本：

```
{
  "asset_type": "audio",
  "call_id": "CALL-445",
  "transcript_text": "My phone is 13800138000 and the gateway upgrade is stuck."
}
```

好样本：

```
{
  "source_id": "support_call",
  "asset_type": "audio",
  "source_system": "call-center",
  "source_fingerprint": "sha256:call-445",
  "schema_version": "v1",
  "owner": "service-ops",
  "access_scope": "restricted_role",
  "pii_level": "sensitive",
  "observed_at": "2026-04-14T09:05:00Z",
  "call_id": "CALL-445",
  "speaker_role": "customer",
  "start_ts": "00:03:12",
  "end_ts": "00:03:40",
  "confidence": 0.94,
  "pii_redaction_flag": false,
  "transcript_text": "My phone is 13800138000 and the gateway upgrade is stuck."
}
```

为什么坏： – 没有 speaker_role / start_ts / end_ts，无法做责任归因和片段定位。 – 没有任何 PII 动作信号，后面只能被动补救。

后果是什么： – 对话审计失效 – HITL 无法回听正确片段 – 高风险字段可能直接进入模型

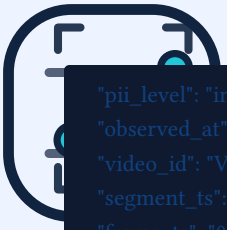
video

坏样本：

```
{
  "asset_type": "video",
  "video_id": "VID-019",
  "subtitle": "Upgrade failed. Retry after checking device compatibility."
}
```

好样本：

```
{
  "source_id": "support_video",
  "asset_type": "video",
  "source_system": "studio-media",
  "source_fingerprint": "sha256:vid-019",
  "schema_version": "v1",
  "owner": "education-content",
  "access_scope": "internal_or_entitled",
```



```
  "pii_level": "internal",
  "observed_at": "2026-04-14T09:05:00Z",
  "video_id": "VID-019",
  "segment_ts": "00:12:05-00:12:40",
  "frame_ts": "00:12:18",
  "transcript_ref": "SEG-778",
  "image_caption": "Firmware upgrade progress screen with retry button",
  "ocr_text": "Upgrade failed. Retry after checking device compatibility."
}
```

为什么坏： – 没有 segment_ts / frame_ts，视频无法按时间和画面稳定回指。 – 只有字幕，没有视觉证据字段，后面难做多模态定位。

后果是什么： – 关键帧证据链断裂 – retrieval 命中后也无法准确落到片段 – 视觉内容和文字内容容易脱节

ticket

坏样本：

```
{
  "asset_type": "ticket",
  "ticket_id": "T-1001",
  "status": "open"
}
```

好样本：

```
{
  "source_id": "ticket_fact",
  "asset_type": "ticket",
  "source_system": "zendesk",
  "source_fingerprint": "sha256:tkt-001",
  "schema_version": "v1",
  "owner": "customer-support-data",
  "access_scope": "tenant+role",
  "pii_level": "sensitive",
  "observed_at": "2026-04-14T09:05:00Z",
  "ticket_id": "T-1001",
  "tenant_id": "tenant-a",
  "status": "open",
  "opened_at": "2026-04-14T08:00:00Z",
  "updated_at": "2026-04-14T09:00:00Z",
  "requester_role": "admin",
  "product_line": "edge-gateway"
}
```

为什么这一步：– 没有 tenant_id / requester_role / product_line，权限和产品线边界都会漂。 – 没有时间字段，后面增量窗口和 SLA 无法建立。

后果是什么： – 多租户过滤不稳 – 增量 ingest 无法准确判断变化范围 – tool boundary 容易错路由

PII 不只看字段，还要看动作

很多团队会在字段清单里只加一列：

- contains_pii = true / false

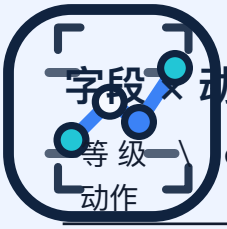
这远远不够。

因为系统真正要做的，不是“知道这里有 PII”，而是：

- 能不能原文存储
- 能不能原文索引
- 能不能送进模型
- 能不能展示给用户
- 失败时该屏蔽、脱敏、隔离还是报警

一套实用的四级 PII 分法

等级	典型内容	默认动作
public	公共帮助文档、公开错误码、公开教程	可索引、可检索、可展示
internal	内部 SOP、非个人化运维说明、内部流程	仅内部角色可用
sensitive	姓名、邮箱、手机号、客户评论、截图中的个人信息	默认脱敏后再进入检索 / 模型
restricted	token、密钥、支付信息、政府证件号、原始敏感音频、强标识符	默认不进索引，不直送模型，必要时隔离或人工审核



字段 × 动作 × 场景：PII 不该只写成 true / false

等级 \ 动作	detect	store_raw	embed	retrieve	display	pass_to_tool	human_review
public	记录即可	允许	允许	允许	允许	允许	可选
internal	记录即可	允许	条件允许	内部范围允许	条件允许	条件允许	建议
sensitive	必须	允许但要标注	默认脱敏后	条件允许	默认裁剪后	默认限制	建议
restricted	必须	条件隔离	默认禁止	默认禁止	默认禁止	默认禁止	必须

先脱敏再入模，还是保真存储后查询裁剪

这不是绝对二选一，而是边界设计问题。

策略	优点	风险
先脱敏再入模	风险更低	信息损失可能影响检索和引用
保真存储 + 查询裁剪	证据链更完整	边界设计更复杂，必须有更强 gate

Week02 推荐的默认判断是：

- sensitive：优先脱敏后消费
- restricted：默认不进入通用 serving

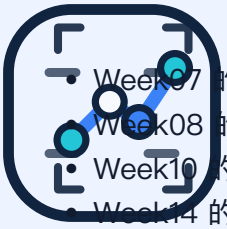
行业新信号｜为什么这件事现在更重要了

1. 多模态检索越来越强调结构保真，而不是“先打平成纯文本再说”。
2. 文档解析越来越强调 hierarchy、layout 和 provenance，因为 citation 和 audit 要求更高。
3. PII 工具可以辅助识别和匿名化，但不等于自动合规；真正的边界仍然要写成动作矩阵和运行时 policy。

这节课和未来多模态系统的关系

今天你写的 metadata，不只影响 Week02。

后面这些能力都会直接复用它：



- Week07 的证据链
- Week08 的 citations 和 retrieval filtering
- Week10 的 tool boundary
- Week14 的 release / governance / audit

所以这节课不是“补备注字段”，而是在定义未来系统的 runtime interface。

直接动手：把标准和样例沉淀进 repo

! 先把这件事说清楚

你今天写的不是说明文，而是后续 gate 会消费的规则。

- metadata_minimums_v1.md 不是注释表，而是 contract 设计前的统一基线。
- pii_policy_matrix_v1.csv 不是备注列表，而是后续 policy / gate 的动作映射。

第 1 步：创建 Week02 蓝图目录和样例目录

```
mkdir -p docs/blueprints/week02
mkdir -p tests/contract/fixtures/week02
```

第 2 步：创建 metadata_minimums_v1.md

```
touch docs/blueprints/week02/metadata_minimums_v1.md
```

第 3 步：创建 pii_policy_matrix_v1.csv

```
touch docs/blueprints/week02/pii_policy_matrix_v1.csv
```

第 4 步：准备 sample_records.json

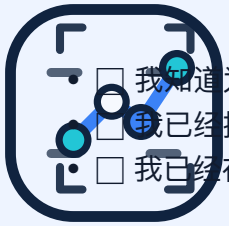
```
touch tests/contract/fixtures/week02/sample_records.json
```

这份样例的目标不是“凑四条记录”，而是：

- 至少 4 条，分别覆盖 ticket / document / audio / video
- 至少 1 条故意带风险，用于下一讲 contract gate 练习

本课自检清单

- ☐ 我知道为什么 metadata 不是备注，而是系统接口
- ☐ 我能说出共享核心 + 模态扩展的统一方法
- ☐ 我已经为四类资产准备了最小 metadata 清单



- □ 我知道为什么 page_no / bbox / section_path / speaker_role / start_ts / frame_ts 是硬约束
- □ 我已经把 PII 写成字段 × 动作矩阵，而不是只有 true / false
- □ 我已经在 repo 里准备好 metadata_minimums_v1.md、pii_policy_matrix_v1.csv、sample_records.json

小结

课时 3 真正解决的，不是“字段更多了”，而是：

从这一讲开始，系统终于知道一条输入到底属于什么、能不能被引用、能不能送进模型、出了问题能不能追责。

课后最小行动

在进入课时 4 之前，请做这 3 件事：

1. 让四类主资产各有一条“好样本”
2. 至少故意保留一条“坏样本”，为下一讲 gate 做准备
3. 选一个你认为风险最高的字段，明确写出它的 pii_level 和默认动作