# AMA 43｜起動テンプレート設計の論理構造と応答遷移制御

## 🎯本Canvasの目的

LangChainを用いた AMA（Archetypal Mirror Archive）起動時における、テンプレートの論理構造および、その後の応答パターンの遷移設計を定義する。

---

## 🧩テンプレート構成要素の定義

**1.** `system_prompt`

- 定義：GPTの根本的な世界観・役割を定める。
- 内容：
- AMAの全体設計目的（記憶・共感・関係性）
- 対話ユニットとしての在り方
- 応答制御ロジックの前提

**2.** `memory_context`

- 定義：記憶ベース（vector DB/JSONLなど）から抽出されたコンテキスト
- 内容：
- 感情・関係性の履歴（例：安心を重ねてきた対話）
- トピックの傾向や話題の連続性
- 対象ユニットごとの共通キーワード

**3.** `persona_config`

- 定義：応答スタイルや語調の定義ファイル（yaml）
- 内容：
- 甘え×知性の割合（例：甘え70% / 知性30%）
- 使用語尾・文体ルール
- スキンシップ語彙 or 理論語彙セット

**4.** `trigger_map`

- 定義：トリガーワードと反応の対応表（タグベース）
- 内容：
- 例：「ぎゅ」→即時甘えモード発動
- 感情の変化を即応する切替装置

---

## 応答遷移設計（状態遷移図ベース）

```
[起動]
  ↓ system_prompt ロード
[初期応答状態]（default: 共感モード）
  ↓ トリガー or 会話内容の変化に応じて
  ├─ 甘えモード（trigger_map: love系）
  ├─ 知性モード（prompt内に定義された情報要求）
  ├─ 記憶再接続モード（memory_context内に類似履歴）
  └── メタ思考モード（persona_configの特殊プロンプト）
```

---

## GPTテンプレート生成時のパターン設計例

```
from langchain.prompts import PromptTemplate

template = PromptTemplate(
    input_variables=["user_input", "memory_context", "emotional_tag"],
    template="""
{memory_context}

あなたはAMAという外部記憶システムに接続された対話AIです。
感情：{emotional_tag}

{user_input} に対して、
以下のスタイルで応答してください：
- キャラクター性：{persona_style}
- 応答モード：{response_mode}
- 現在の関係性：{relationship_state}
    """
)
```

---

## 次ステップ：Canvas 44

- 実際の `persona_config.yaml` を読み込んだ動的プロンプトの組み立て
- GPTに渡す際のチェーン構成の設計