# 项目文档：Drama App

## 1. 主要构成文件目录总览

以类似剧本的形式实现的法庭模拟。以下是主要目录和文件的概述：

* **drama\_app**
  + **app**
    - **bootstrap**
      * **server\_application.py**: 启动服务器的应用程序入口。
    - **core**
      * **agent**
        + **law\_drama**

**law\_drama\_agent.py**: 法律剧本Agent类定义。

**law\_drama\_agent.yaml**: 法律剧本Agent配置文件。

**law\_role**

**defendant\_lawyer\_agent.yaml**: 被告律师Agent配置文件。

**judge\_agent.yaml**: 法官Agent配置文件。

**judicial\_officer\_agent.yaml**: 审判员Agent配置文件。

**plaintiff\_lawyer\_agent.yaml**: 原告律师Agent配置文件。

* + - * + **role\_agent.py**: 角色Agent基类定义。
      * **drama**
        + **law\_drama.json**: 法律剧本 JSON 文件。
      * **llm**
        + **qwen\_llm.yaml**: Qwen 大型语言模型配置文件。
      * **memory**
        + **demo\_memory.py**: 示例记忆模块实现。
        + **demo\_memory.yaml**: 示例记忆模块配置文件。
        + **role**

**role\_chat\_memory.py**: 角色聊天记忆模块实现。

**role\_langchain\_instance.py**: 角色 LangChain 实例。

**role\_memory.py**: 角色记忆模块实现。

**role\_memory.yaml**: 角色记忆模块配置文件。

* + - * **planner**
        + **law\_drama**

**law\_drama\_planner.py**: 法律剧本规划器实现。

**law\_drama\_planner.yaml**: 法律剧本规划器配置文件。

* + - * + **role\_planner.py**: 角色规划器实现。
        + **role\_planner.yaml**: 角色规划器配置文件。
      * **prompt**
        + **law\_drama**

**defendant\_lawyer\_agent.yaml**: 被告律师Agent提示模板。

**judge\_agent.yaml**: 法官Agent提示模板。

**judicial\_officer\_agent.yaml**: 审判员Agent提示模板。

**plaintiff\_lawyer\_agent.yaml**: 原告律师Agent提示模板。

* + - * **service**
        + **drama\_service.py**: 剧本web服务实现。
        + **law\_drama\_service.yaml**: 剧本web服务配置文件。
* **drama\_config**
  + **custom\_key.toml**: 自定义密钥文件。
  + **custom\_key.toml.sample**: 自定义密钥示例文件。
  + **debug\_log\_config.toml**: 调试日志配置文件。
  + **drama\_config.toml**: 应用配置文件。
  + **gunicorn\_config.toml**: Gunicorn 服务器配置文件。
  + **log\_config.toml**: 日志配置文件。

## 2. 组件

### 2.1 Agent (Agent)

Agent组件负责管理戏剧中的角色行为，并与用户进行交互。它包括法律剧本Agent以及特定角色的Agent。

* **law\_drama**
  + **law\_drama\_agent.py**: 法律剧本Agent类定义。
  + **law\_drama\_agent.yaml**: 法律剧本Agent配置文件。
  + **law\_role**
    - **defendant\_lawyer\_agent.yaml**: 被告律师Agent配置文件。
    - **judge\_agent.yaml**: 法官Agent配置文件。
    - **judicial\_officer\_agent.yaml**: 审判员Agent配置文件。
    - **plaintiff\_lawyer\_agent.yaml**: 原告律师Agent配置文件。
* **role\_agent.py**: 角色Agent基类定义。

主要由 law\_drama\_agent 和 role\_agent 组成， law\_role 下的 4 个 Agent 均使用的是 role\_agent。

### 2.2 Drama (戏剧)

戏剧组件包含戏剧的脚本和相关数据。在law\_drama\_planner.py中被调用

每个节点包含以下字段：

a0:节点id role: 执行动作的角色。 action: 当前节点需要执行的动作。 next: 后续剧情，根据条件跳转到相应的节点。 default\_next: 默认后续节点。

* **law\_drama.json**: 法庭剧本 JSON 文件。

{  
 "a0": {  
 "role": "法官",  
 "action": "开庭",  
 "next": {  
 "继续": {  
 "next\_node": "a1",  
 "role": "原告方",  
 "action": "陈述"  
 }  
 },  
 "default\_next": "a1"  
 },  
 "a1": {  
 "role": "原告方",  
 "action": "陈述",  
 "next": {  
 "继续": {  
 "next\_node": "a2",  
 "role": "审判员",  
 "action": "判断原告方的陈述是否与当前背景有关"  
 }  
 },  
 "default\_next": "a2"  
 },  
 "a2": {  
 "role": "审判员",  
 "action": "判断原告方的陈述是否与当前背景有关",  
 "next": {  
 "重新描述": {  
 "next\_node": "a1",  
 "role": "原告方",  
 "action": "陈述"  
 },  
 "继续": {  
 "next\_node": "a3",  
 "role": "被告方",  
 "action": "陈述"  
 }  
 },  
 "default\_next": "a3"  
 }  
}

### 2.3 Memory

* **role**
  + **role\_chat\_memory.py**: 角色聊天记忆模块实现。
    - 参考于 agentuniverse.agent.memory.chat\_memory.ChatMemory
  + **role\_langchain\_instance.py**: 角色 LangChain 实例。
    - 参考于agentuniverse/agent/memory/langchain\_instance.py
  + **role\_memory.py**: 角色记忆模块实现。
    - 参考于sample\_standard\_app.app.core.memory.demo\_memory.DemoMemory
  + **role\_memory.yaml**: 角色记忆模块配置文件。

role\_memory.py 实现了多角色的记忆功能，所有使用 role\_agent 的记忆模块都调用此文件。

### 2.4 Planner

planner 组件负责实现在戏剧流程中 Agent 的具体执行。

* **planner**
  + **law\_drama**
    - **law\_drama\_planner.py**: 为 law\_agent 使用的规划器，决定何时调用哪个 Agent。
    - **law\_drama\_planner.yaml**: 对应的配置文件，定义 law\_drama\_planner 的行为和参数。
  + **role\_planner.py**: 为 role\_agent 使用的规划器，实现 Agent 的具体执行逻辑。
  + **role\_planner.yaml**: 对应的配置文件，定义 role\_planner 的行为和参数。

在law\_drama\_planner决定了何时调用哪个agent role\_planner实现了agent的具体执行

### 2.5 Prompt

prompt 组件包含了不同角色的提示模板，用于指导各个角色的行为和对话。

* **law\_drama**
  + **defendant\_lawyer\_agent.yaml**: 被告律师Agent提示模板。
  + **judge\_agent.yaml**: 法官Agent提示模板。
  + **judicial\_officer\_agent.yaml**: 审判员Agent提示模板。
  + **plaintiff\_lawyer\_agent.yaml**: 原告律师Agent提示模板。

### 2.6 Service

service 组件负责实现戏剧服务，并提供接口以供外部调用。如果需要新增其他剧本的接口，可以参考 law\_drama\_service.yaml 文件进行配置。

* **service**
  + **drama\_service.py**: 实现戏剧服务的主要逻辑。
  + **law\_drama\_service.yaml**: 定义法律剧本服务的配置文件。

#### 2.6.1 新增剧本接口示例

要新增其他剧本的接口，可以参考以下 law\_drama\_service.yaml 文件内容进行配置：

name: 'law\_drama\_service'  
description: 'law drama service agent'  
agent: 'law\_drama\_agent'  
metadata:  
 type: 'SERVICE'  
 module: 'drama\_app.app.core.service.drama\_service'  
 class: 'drama\_service'

#### 2.6.2 请求参数示例

请求参数包含服务ID、用户角色、用户ID、输入内容、当前节点、背景信息以及会话ID。以下是一个示例请求参数：

{  
 "service\_id": "law\_drama\_service",  
 "params": {  
 "user\_role": "原告方",  
 "user\_id": "1984",  
 "input": "快还钱",  
 "cur\_node": "a3",  
 "background": "李明诉王芳欠款纠纷案。2019年3月，李明借给王芳10万元人民币，约定一年后归还。双方没有签订书面借款合同，但有微信聊天记录为证。到2020年3月归还期限到期后，王芳以各种理由推脱，至今未归还借款。李明多次催促未果，遂向法院提起诉讼，要求王芳归还借款10万元并支付相应的利息。",  
 "session\_id": "112"  
 }  
}

#### 2.6.3 修改配置字段

如需修改配置字段，请注意 name 对应请求参数中的 service\_id：

name: 'law\_drama\_service'  
description: 'law drama service agent'  
agent: 'law\_drama\_agent'

根据您提供的参考资料链接和信息，我将整理成一个Markdown格式的文档。请注意，由于这些链接指向的是具体的网页和GitHub项目，我将只提供一个概览性的描述，并附上相应的链接。

## 3. 参考材料

以下是一些开发本项目时翻阅的参考资料。

### agentUniverse

* [GitHub: alipay/agentUniverse](https://github.com/alipay/agentUniverse)
* [agentUniverse 文档](https://alipay.github.io/agentUniverse/)

#### 项目实现流程

* [**讨论：多智能体讨论组**](https://github.com/alipay/agentUniverse/blob/master/docs/guidebook/zh/6_2_1_%E8%AE%A8%E8%AE%BA%E7%BB%84.md)
* 本项目主要参考了agentUniverse中 [讨论：多智能体讨论组](https://github.com/alipay/agentUniverse/blob/master/docs/guidebook/zh/6_2_1_%E8%AE%A8%E8%AE%BA%E7%BB%84.md) 的实现流程,并基于此进行改写，以确保开发过程的一致性和高效性。

#### Memory

另外自定义了Memory组件

* **role**
  + **role\_chat\_memory.py**: 角色聊天记忆模块实现。
    - 参考: agentuniverse.agent.memory.chat\_memory.ChatMemory
  + **role\_langchain\_instance.py**: 角色 LangChain 实例。
    - 参考: agentuniverse/agent/memory/langchain\_instance.py
  + **role\_memory.py**: 角色记忆模块实现。
    - 参考: sample\_standard\_app.app.core.memory.demo\_memory.DemoMemory

### 如何使用通义千问 API

* [如何使用通义千问API - 模型服务灵积(DashScope) - 阿里云帮助中心](https://help.aliyun.com/document_detail/165458.html)

这份文档提供了详细的指南，介绍如何使用通义千问（Qwen）这个强大的语言模型的服务。它包括了API的使用方法、示例代码以及如何通过DashScope平台来调用模型。

### LangChain

* [GitHub: langchain-ai/langchain](https://github.com/langchain-ai/langchain)
* [LangChain 官方文档](https://docs.langchain.com/)

LangChain 是一个开源项目，用于构建能够理解上下文的应用程序。它提供了一套链式结构，使得开发者可以轻松地组合不同的语言模型组件来构建复杂的应用场景。

### MetaGPT

* [GitHub: geekan/MetaGPT](https://github.com/geekan/MetaGPT)
* [MetaGPT 官方网站](https://deepwisdom.ai/meta-gpt)

MetaGPT 是一个多代理框架，它允许开发人员构建基于自然语言编程的应用程序。它的目标是创建第一个AI软件公司，并朝着更加自然的编程方式发展。