# 测试文档：仅针对新增的异步接口与 Batch 批处理

以下步骤假设你已经在项目根目录 `Project_raccoon` 中，并且 `.env` 里的 `GOOGLE_API_KEY` 已配置可用。

---

## 一、环境与服务启动

1. **安装依赖**

```
cd C:\Users\Lenovo\Desktop\Project_raccoon
pip install -r requirements.txt
```

2. **启动 FastAPI 服务**

```
uvicorn src.api.server:app --host 0.0.0.0 --port 8000 --reload
```

- 正常情况下终端会显示 Uvicorn 启动日志。
- 后续所有 HTTP 测试默认访问 `http://127.0.0.1:8000`。

---

## 二、单任务异步接口测试（/api/v1/jobs/image）

### 2.1 准备测试图片

- 确保 `input/` 目录下有至少一张图片，例如：`input/test_fragment2.png`。
- 你也可以随意新放一张 `PNG/JPG/JPEG` 图片。

### 2.2 提交异步任务

在 PowerShell / CMD 中执行（注意改为你实际的图片路径）：

```
curl -X POST "http://127.0.0.1:8000/api/v1/jobs/image" ^
  -F "file=@input/test_fragment2.png"
```

预期返回类似：

```
{"task_id": "c1b6f6c4-1234-...."}
```

记录下 `task_id`。

### 2.3 轮询任务状态

用上一步的 `task_id`：

```
curl "http://127.0.0.1:8000/api/v1/jobs/c1b6f6c4-1234-...."
```

你会看到类似结构：

```
{
  "task_id": "c1b6f6c4-1234-....",
  "status": "RUNNING",
  "created_at": "...",
  "updated_at": "...",
  "result": null,
  "error": null
}
```

- 多等几秒后再次请求，直到 `status` 变为 `SUCCEEDED` 或 `FAILED`。
- 当为 `SUCCEEDED` 时，`result` 字段中是完整的 `FinalAnswer` JSON。

### 2.4 验证本地输出与轮次记录

任务成功后，检查以下文件是否出现（文件名前缀为图片名，无扩展名）：

- `output/<图片名>_result.json`
- `output/<图片名>_report.txt`
- `output/<图片名>_note.txt`
- `sessions/<session_id>.rounds.jsonl`

其中：

- `*_result.json`：结构化结果，结构应与 `src/schemas.py` 中 `FinalAnswer` 一致。
- `*_report.txt` / `*_note.txt`：说明性文本。
- `*.rounds.jsonl`：每行一个轮次摘要，包含：
  - `round_index`
  - `summary`
  - `tool_calls`（工具名、参数、结果摘要）
  - `notes`

这一步主要验证：**异步接口确实复用了原有多轮推理与本地持久化逻辑**。

---

## 三、Batch 批处理接口测试（/api/v1/batches）

### 3.1 准备多张测试图片

确保 `input/` 下至少有 2–3 张图片，例如：

- `input/test_fragment2.png`
- `input/test_fragment3.png`

### 3.2 创建批处理任务（多文件上传）

```
curl -X POST "http://127.0.0.1:8000/api/v1/batches" ^
  -F "files=@input/test_fragment2.png" ^
  -F "files=@input/test_fragment3.png"
```

预期返回：

```
{"batch_id": "e9d5...."}
```

记录下 `batch_id`。

### 3.3 轮询批处理状态

```
curl "http://127.0.0.1:8000/api/v1/batches/e9d5...."
```

返回示例：

```
{
  "batch_id": "e9d5....",
  "status": "BATCH_RUNNING",
  "round": 1,
  "total_jobs": 2,
  "completed_jobs": 0,
  "failed_jobs": 0,
  "details": [
    {
      "session_id": "...",
      "alias": "test_fragment2_xxxxxxxx",
      "done": false,
      "error": null,
      "last_round": 0
    },
    ...
  ]
}
```

- 多等几轮（间隔 10–20 秒）重复请求，直到：
  - `status` 变为 `SUCCEEDED` 或 `FAILED`；

- `completed_jobs` + `failed_jobs` == `total_jobs`。

### 3.4 查看批处理结果

1. **查看整体结果**

```
curl "http://127.0.0.1:8000/api/v1/batches/e9d5..../results"
```

预期结构：

```
{
  "batch_id": "e9d5....",
  "items": [
    {
      "session_id": "...",
      "status": "SUCCEEDED",
      "result": { ... FinalAnswer ... },
      "error": null
    },
    ...
  ]
}
```

2. **查看单个 session 结果**

从 `items` 里拿一个 `session_id`，再调用：

```
curl "http://127.0.0.1:8000/api/v1/batches/e9d5..../results?session_id=<某个session_id>"
```

验证：

- `status` 为 `SUCCEEDED` 时，`result` 中为完整的 `FinalAnswer`；
- 若为 `FAILED`，`error` 字段里应有失败原因（如 “模型未返回结构化结果”、“最终结构化输出失败”等）。

### 3.5 检查本地文件与轮次记录

Batch 成功后，对每张图片应看到：

- `output/<alias>_result.json`
- `output/<alias>_report.txt`
- `output/<alias>_note.txt`
- 对应 `sessions/<session_id>.rounds.jsonl`

说明：

- `<alias>` 默认是“图片文件名 + 下划线 + session_id 前 8 位”，例如：`test_fragment2_1a2b3c4d`。
- 打开 `*.rounds.jsonl`，可以看到多个 `round_index`，并且 `tool_calls` 中应该包含批处理中 AI 决定调用的 CBETA/Gallica 工具及其结果摘要。
- 这一步验证：**Batch 模式下依然是“多轮调用工具 + 本地持久化”，只是每一轮是批量通过 Gemini Batch API 发出。**

---

## 四、简单回归检查（确认未破坏旧 CLI 流程）

虽不在“新增内容”范围，但建议做一个极简回归：

```
python -m src.main --input input --output output
```

确认：

- 仍然能逐张处理 `input/` 中的图片；
- 输出的 `*_result.json`、`*_report.txt`、`*_note.txt` 结构未变化；
- `sessions/` 下仍有新的 rounds 记录产生。

---

## 五、常见问题与排查思路（仅针对新增部分）

- **问题 1：访问接口报 422 / 400**
  - 检查是否真的用 `-F "file=@...png"` 或 `-F "files=@...png"`，而不是 `-d`。
- **问题 2：状态一直停在 RUNNING / BATCH_RUNNING**
  - 查看服务端终端是否有 Google API 报错（429 / 503 等）。
  - 检查 `.env` 中 `GOOGLE_API_KEY` 是否正确、网络是否可访问 Gemini。
- **问题 3：Batch 报 JOB_STATE_FAILED**
  - 说明 Batch API 侧整体失败，可在代码里临时打印 `job.error` 进一步诊断，或对照官方 Batch API 文档 `batchStats` 与错误码说明。

按照以上步骤走一遍，你就能比较系统地确认：

- 单图异步接口是否工作正常；
- 多图 Batch 管线是否按“多轮对话 + 本地工具调用 + 会话持久化”的预期运转。