📘 AT 命令文档生成流水线

从 CSV 到 HTML/PDF 的自动化解决方案

🔷 Step 0: 环境初始化

安装必要依赖包（首次运行时执行）

```
# %%
!pip install pandas pyyaml sphinx sphinx_rtd_theme --quiet
```

🔷 Step 1: 准备 CSV 数据文件

先创建示例数据文件 `commands.csv`（实际使用时替换为真实数据）

```
# %%
import pandas as pd
# 创建示例数据
data = {
    "命令": ["AT+CREG", "AT+CWJAP"],
    "命令标题": ["查询网络注册状态", "连接WiFi网络"],
    "命令类型": ["执行;查询;测试", "设置;查询"],
    "命令格式": ["AT+CREG=[<n>]|AT+CREG?|AT+CREG=?", "AT+CWJAP=<ssid>,<pwd>
[,<bssid>][,<prio>]|AT+CWJAP?"],
    "示例命令": ["AT+CREG=1|AT+CREG?|AT+CREG=?",
'AT+CWJAP="MyWiFi","123456"|AT+CWJAP?'],
    "示例响应": ["OK|+CREG: 0,1 OK|+CREG: (0-2) OK",
'OK|+CWJAP:"MyWiFi","aa:bb:cc:dd:ee:ff",-50,1 OK'],
    "功能描述": ["查询或控制模块的网络注册状态", "连接到指定WiFi网络"],
    "备注": ["AT+CREG=5 返回 ERROR（参数超出范围）", "密码需为8-64字节ASCII字
符"],
    "参数JSON": [
        # 正确格式：values是字典
        '[{"name":"<n>","desc":"控制结果代码输出方式","values":{"0":"禁用上
报","1":"启用上报","2":"上报并包含LAC/CI信息"}},{"name":"<stat>","desc":"网络注
册状态","values":{"0":"未注册","1":"已注册本地网络","2":"正在搜索"}}]',
        # 修复WiFi参数的values格式
        '[{"name":"<ssid>","desc":"WiFi名称","values":{"格式":"字符串","长
度":"1-32字节"}},{"name":"<pwd>","desc":"WiFi密码","values":{"格式":"ASCII字
符","长度":"8-64字节"}}]'
    ]
}
# 保存为CSV文件
```

```
pd.DataFrame(data).to_csv("commands.csv", index=False, encoding="utf-8")
print("✅ 已生成示例数据文件: commands.csv")
```

```
✅ 已生成示例数据文件: commands.csv
```

🔷 Step 2: CSV 转 YAML/JSON

将表格数据转换为结构化配置文件

```
# %%
import pandas as pd
import json
import yaml
import os
# 创建数据目录
os.makedirs("data", exist_ok=True)
# 读取CSV文件
df = pd.read_csv("commands.csv")
cmd_objects = []
# 转换为结构化数据
for _, row in df.iterrows():
    # 解析参数JSON
    param_json = json.loads(row["参数JSON"])

    # 构建命令对象
    cmd_data = {
        "command": row["命令"],
        "title": row["命令标题"],
        "type": [t.strip() for t in row["命令类型"].split(";")],
        "formats": [f.strip() for f in row["命令格式"].split("|")],
        "parameters": param_json,
        "examples": [
            {"cmd": c.strip(), "resp": r.strip()}
            for c, r in zip(row["示例命令"].split("|"), row["示例响
应"].split("|"))
        ],
        "description": row["功能描述"],
        "notes": row["备注"]
    }
    cmd_objects.append(cmd_data)

    # 保存单个命令JSON
    with open(f"data/{row['命令']}.json", "w", encoding="utf-8") as f:
        json.dump(cmd_data, f, ensure_ascii=False, indent=2)
# 保存所有命令YAML
with open("all_commands.yaml", "w", encoding="utf-8") as f:
    yaml.safe_dump({"commands": cmd_objects}, f, allow_unicode=True,
sort_keys=False)
```

```
print(f"✅ 生成 {len(cmd_objects)} 个命令数据文件:")
print(f"- 单个命令JSON: data/*.json")
print(f"- 汇总YAML: all_commands.yaml")
```

```
✅ 生成 2 个命令数据文件:
- 单个命令JSON: data/*.json
- 汇总YAML: all_commands.yaml
```

🔷 Step 3: YAML 转 reStructuredText (RST)

生成 Sphinx 所需的结构化文档源文件

```
# %%
import yaml
import os
# 创建输出目录
os.makedirs("rst_output", exist_ok=True)
# 加载YAML数据
with open("all_commands.yaml", "r", encoding="utf-8") as f:
    commands = yaml.safe_load(f)["commands"]
# RST生成函数
# 修改 Step 3 中的 generate_command_rst 函数，添加参数验证
def generate_command_rst(cmd):
    rst = [
        f"{cmd['command']}\n",
        f"{'=' * len(cmd['command'])}\n\n",
        f"**命令标题**: {cmd['title']}\n\n",
        f"**命令类型**: {', '.join(cmd['type'])}\n\n",
        "命令格式::\n"
    ]
    for fmt in cmd["formats"]:
        rst.append(f"  {fmt}\n")

    # 参数说明表格（添加错误处理）
    rst.append("\n参数说明\n--------\n")
    rst.append(".. list-table::\n")
    rst.append("   :header-rows: 1\n")
    rst.append("   :widths: 15 30 45\n\n")
    rst.append("   * - 参数名\n     - 描述\n     - 取值范围\n")

    for p in cmd["parameters"]:
        # 验证 values 类型并处理
        if isinstance(p["values"], dict):
            # 正常字典格式
            values = "\n".join([f"{k}: {v}" for k, v in
p["values"].items()])
        elif isinstance(p["values"], str):
```

```
            # 字符串格式（添加警告提示）
            values = p["values"] + " ⚠ [格式警告：应为键值对]"
        else:
            values = "格式错误"

        rst.append(f"   * - {p['name']}\n     - {p['desc']}\n     -
{values}\n")

    # 示例代码块（保持不变）
    rst.append("\n示例\n----\n")
    for ex in cmd["examples"]:
        rst.append(".. code-block:: none\n\n")
        rst.append(f"   命令: {ex['cmd']}\n")
        rst.append(f"   响应: {ex['resp']}\n\n")

    # 功能描述和备注（保持不变）
    rst.append(f"**功能描述**：{cmd['description']}\n\n")
    if cmd["notes"]:
        rst.append(f"**注意事项**：{cmd['notes']}\n")

    return "".join(rst)
# 生成命令RST文件
for cmd in commands:
    rst_content = generate_command_rst(cmd)
    with open(f"rst_output/{cmd['command']}.rst", "w", encoding="utf-8")
as f:
        f.write(rst_content)
# 生成索引RST
index_rst = [
    "AT 命令手册\n",
    "==================\n\n",
    ".. toctree::\n",
    "   :maxdepth: 2\n",
    "   :caption: 命令列表\n\n"
]
for cmd in commands:
    index_rst.append(f"   {cmd['command']}\n")
with open("rst_output/index.rst", "w", encoding="utf-8") as f:
    f.write("".join(index_rst))
print(f"✅ 生成 RST 文档:")
print(f"- 命令文档: rst_output/*.rst")
print(f"- 索引文件: rst_output/index.rst")
```

```
✅ 生成 RST 文档:
- 命令文档: rst_output/*.rst
- 索引文件: rst_output/index.rst
```

🔷 Step 4: 初始化 Sphinx 项目

创建文档工程并配置主题

```
# %%
# 初始化Sphinx项目
!sphinx-quickstart docs --sep --project "AT Command Manual" --author "有方科技" --release "1.0" -q
# 修改配置文件（conf.py）
import fileinput
import os
conf_path = "docs/source/conf.py"
lines = []
with open(conf_path, "r", encoding="utf-8") as f:
    lines = f.readlines()
# 更新配置内容
new_lines = []
for line in lines:
    if line.startswith("html_theme ="):
        new_lines.append("html_theme = 'sphinx_rtd_theme'\n")
    elif line.startswith("import sys"):
        new_lines.append(line)
        new_lines.append("import os\n")
        new_lines.append("sys.path.insert(0, os.path.abspath('../..'))\n")
    else:
        new_lines.append(line)
with open(conf_path, "w", encoding="utf-8") as f:
    f.writelines(new_lines)
print("✅ Sphinx项目初始化完成:")
print("- 项目路径: docs/")
print("- 配置文件: docs/source/conf.py")
```

```
 [01mFinished: An initial directory structure has been created. [39;49;00m

You should now populate your master file
/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/index.rst and create other documentation
source files. Use the Makefile to build the docs, like so:
   make builder
where "builder" is one of the supported builders, e.g. html, latex or
linkcheck.

✅ Sphinx项目初始化完成:
- 项目路径: docs/
- 配置文件: docs/source/conf.py
```

🔷 Step 5: 构建 HTML 文档

将 RST 文件编译为可浏览的 HTML 格式

```
# %%
# 复制RST文件到Sphinx源目录
!cp -r rst_output/* docs/source/
```

```
# 构建HTML文档
!make -C docs html
print("\n✅ HTML文档构建完成:")
print(f"打开查看: file://{os.getcwd()}/docs/build/html/index.html")
```

```
[01mRunning Sphinx v8.2.3[39;49;00m
[01mloading translations [en]... [39;49;00mdone
[01mmaking output directory... [39;49;00mdone
[01mbuilding [mo]: [39;49;00mtargets for 0 po files that are out of date
[01mwriting output... [39;49;00m
[01mbuilding [html]: [39;49;00mtargets for 3 source files that are out of
date
[01mupdating environment: [39;49;00m[new config] 3 added, 0 changed, 0
removed
[2K[01mreading sources... [39;49;00m[100%] [35mindex[39;49;00m[00m
[91m/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/AT+CREG.rst:25: WARNING: Explicit markup ends without a
blank line; unexpected unindent. [docutils][39;49;00m
[31m/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/AT+CREG.rst:27: ERROR: Unexpected indentation.
[docutils][39;49;00m
[91m/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/AT+CREG.rst:30: WARNING: Block quote ends without a blank
line; unexpected unindent. [docutils][39;49;00m
[91m/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/AT+CWJAP.rst:24: WARNING: Explicit markup ends without a
blank line; unexpected unindent. [docutils][39;49;00m
[91m/Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/source/AT+CWJAP.rst:28: WARNING: Definition list ends without a
blank line; unexpected unindent. [docutils][39;49;00m
[01mlooking for now-outdated files... [39;49;00mnone found
[01mpickling environment... [39;49;00mdone
[01mchecking consistency... [39;49;00mdone
[01mpreparing documents... [39;49;00mdone
[01mcopying assets... [39;49;00m
[01mcopying static files... [39;49;00m
Writing evaluated template result to /Users/pika/Documents/GitHub/docs-as-
code-learning/demo-1006/docs/build/html/_static/basic.css
Writing evaluated template result to /Users/pika/Documents/GitHub/docs-as-
```

```
code-learning/demo-1006/docs/build/html/_static/language_data.js
Writing evaluated template result to /Users/pika/Documents/GitHub/docs-as-
code-learning/demo-1006/docs/build/html/_static/documentation_options.js
Writing evaluated template result to /Users/pika/Documents/GitHub/docs-as-
code-learning/demo-1006/docs/build/html/_static/js/versions.js
 [01mcopying static files:  [39;49;00mdone
 [01mcopying extra files...  [39;49;00m
 [01mcopying extra files:  [39;49;00mdone
 [01mcopying assets:  [39;49;00mdone
 [2K [01mwriting output...  [39;49;00m[100%]  [32mindex [39;49;00m00m
 [01mgenerating indices...  [39;49;00mgenindex done
 [01mwriting additional pages...  [39;49;00msearch done
 [01mdumping search index in English (code: en)...  [39;49;00mdone
 [01mdumping object inventory...  [39;49;00mdone
 [01mbuild succeeded, 5 warnings. [39;49;00m

The HTML pages are in build/html.

✅ HTML文档构建完成:
打开查看: file:///Users/pika/Documents/GitHub/docs-as-code-learning/demo-
1006/docs/build/html/index.html
```

🔷 Step 6 (可选): 构建 PDF 文档

需要安装TeXLive环境（仅Linux环境支持）

```
# %%
# # 安装依赖（仅首次运行）
# !sudo apt-get install texlive-xetex -y --quiet
```

```
# # 构建PDF文档
# !make -C docs latexpdf
# print("\n✅ PDF文档构建完成:")
# print(f"文件路径: docs/build/latex/atcommandmanual.pdf")
```