PowerAutomation Playwright 集成系統完整文檔

概述

PowerAutomation Playwright集成系統是一個完整的自動化解決方案，實現了直接操作Manus頁面的高級功能。通過Playwright瀏覽器自動化技術，系統能夠自動登錄Manus、遍歷任務列表、獲取完整對話歷史、分類下載文件，並提供智能介入分析功能。

系統架構

核心組件

1. Manus高級控制器 (ManusAdvancedController)

- **文件位置**: `/home/ubuntu/smartinvention/shared/manus_advanced_controller.py`
- **主要功能**:
- 自動登錄Manus系統
- 遍歷左側任務列表
- 獲取完整對話歷史
- 文件分類和批量下載
- 智能滾動載入更多歷史

2. Flask API服務器

- **文件位置**: `/home/ubuntu/smartinvention/ec2/powerautomation_manus_api/`
- **主要功能**:
- 提供RESTful API接口
- 集成Manus控制器
- 支持跨域請求(CORS)
- 智能介入分析

3. API路由模塊

- **文件位置**: `/home/ubuntu/smartinvention/ec2/powerautomation_manus_api/src/routes/manus.py`
- **主要功能**:
- 服務啟動/停止控制
- 消息發送和接收
- 對話分析和介入
- 系統狀態監控

核心功能詳解

1. 任務列表遍歷

系統能夠自動識別和遍歷Manus左側的任務列表，就像TRAE的倉庫列表一樣。每個任務對應一個特定的對話主題或項目。

功能特點: - 自動檢測任務列表容器 - 提取任務標題、ID和最後活動時間 - 支持點擊導航到特定任務 - 動態載入任務信息

技術實現:

```
async def get_task_list(self) -> List[TaskInfo]:
    # 查找任務列表容器
    task_list_selectors = [
        '.task-list', '.conversation-list', '.sidebar-list',
        '[data-testid="task-list"]', '.chat-list'
    ]

    # 提取任務信息
    for element in task_elements:
        task_info = TaskInfo(
            task_id=task_id,
            title=title.strip(),
            url=href,
            last_activity=last_activity
        )
```

2. 文件分類和批量下載

系統實現了完整的文件管理功能，支持按類型分類和批量下載。

支持的文件分類: - **Documents**: PDF、DOC、TXT、MD等文檔文件 - **Images**: PNG、JPG、GIF、SVG等圖片文件
- **Code files**: PY、JS、HTML、CSS等代碼文件 - **Links**: 外部鏈接和URL

批量下載流程: 1. 點擊"View all files in this task"按鈕 2. 選擇文件分類標籤（紅色圈圈） 3. 點擊批量下載按鈕（綠色圈圈） 4. 自動創建分類目錄並下載文件

技術實現:

```
 async def batch_download_files(self, category: str = "All") -> Dict[str,
List[FileInfo]]:
    # 打開文件面板
    await self.open_files_panel()

    # 按分類獲取文件
    for cat in ["Documents", "Images", "Code files", "Links"]:
        files = await self.get_files_by_category(cat)

    # 批量下載
    for file_info in files:
        await self.download_file(file_info)
```

3. 完整對話歷史獲取

系統能夠獲取任務的完整對話歷史，包括通過滾動載入的歷史消息。

功能特點: - 自動滾動載入更多歷史 - 識別用戶和AI消息 - 提取時間戳和消息內容 - 支持HTML格式保存

技術實現:

```
async def get_conversation_history(self, task_info: TaskInfo) -> List[Dict[str, Any]]:
    # 導航到任務
    await self.navigate_to_task(task_info)

    # 滾動載入歷史
    await self.scroll_to_load_history()

    # 獲取所有消息
    messages = await self.get_all_messages()
    return messages
```

4. 智能介入分析

系統提供智能分析功能，自動判斷是否需要介入對話。

分析維度: - 對話複雜度評估 - 關鍵詞匹配分析 - 介入優先級判斷 - 智能建議生成

API接口文檔

基礎接口

健康檢查

```
GET /api/manus/health
```

響應示例:

```
{
  "service": "PowerAutomation Manus Integration",
  "status": "healthy",
  "manus_service_running": true,
  "timestamp": "2025-06-22T13:40:00Z"
}
```

啟動服務

```
POST /api/manus/start
```

響應示例:

```
{
  "success": true,
  "message": "Manus服務啟動成功",
  "status": "started"
}
```

停止服務

```
POST /api/manus/stop
```

核心功能接口

發送消息

```
POST /api/manus/send_message
```

請求體:

```
{
  "message": "您的消息內容",
  "context": {
    "repository": "communitypowerauto",
    "conversation_id": "conv_001"
  }
}
```

響應示例:

```
{
  "success": true,
  "message_sent": "您的消息內容",
  "response_received": "Manus的回覆",
  "latest_messages": [...],
  "timestamp": "2025-06-22T13:40:00Z"
}
```

獲取對話狀態

```
GET /api/manus/get_conversation
```

響應示例:

```
{
  "success": true,
  "is_logged_in": true,
  "page_info": {
    "url": "https://manus.chat/task/123",
    "title": "Smart Invention Project"
  },
  "recent_messages": [...],
  "message_count": 15
}
```

分析對話

```
POST /api/manus/analyze_conversation
```

請求體:

```
{
  "messages": [
    {"role": "user", "content": "用戶消息"},
    {"role": "manus", "content": "Manus回覆"}
  ],
  "repository": "communitypowerauto",
  "conversation_id": "conv_001"
}
```

響應示例:

```
{
  "success": true,
  "analysis": {
    "intervention_needed": true,
    "confidence": 0.85,
    "priority": "high",
    "suggestion": "建議進行智能介入"
  }
}
```

智能介入

```
POST /api/manus/intelligent_intervention
```

請求體:

```
{
  "messages": [...],
  "intervention_type": "suggestion",
  "context": {
    "repository": "communitypowerauto"
  }
}
```

部署和配置

環境要求

- Python 3.11+
- Playwright
- Flask
- Chromium瀏覽器

安裝步驟

1. **創建虛擬環境:**

```
cd /home/ubuntu/smartinvention/ec2/powerautomation_manus_api
python -m venv venv
source venv/bin/activate
```

1. **安裝依賴:**

```
pip install -r requirements.txt
playwright install chromium
```

1. **啟動服務**:

```
python src/main.py
```

配置選項

Manus登錄憑證

```
credentials = {
    'email': 'chuang.hsiaoyen@gmail.com',
    'password': 'silentfleet#1234'
}
```

瀏覽器配置

```
config = {
    'url': 'https://manus.chat',
    'headless': True,         # 無頭模式
    'slow_mo': 1000,          # 操作間隔
    'timeout': 60000          # 超時時間
}
```

使用示例

Python客戶端示例

```
import requests
import json

# 啟動Manus服務
response = requests.post('http://localhost:5000/api/manus/start')
print(response.json())

# 發送消息
message_data = {
    'message': '🧪 PowerAutomation測試消息',
    'context': {
        'repository': 'communitypowerauto',
        'conversation_id': 'conv_test_001'
    }
}

response = requests.post(
    'http://localhost:5000/api/manus/send_message',
    json=message_data
)
print(response.json())

# 獲取對話狀態
response = requests.get('http://localhost:5000/api/manus/get_conversation')
print(response.json())
```

JavaScript客戶端示例

```javascript
// 啟動服務
async function startManusService() {
    const response = await fetch('/api/manus/start', {
        method: 'POST'
    });
    return await response.json();
}

// 發送消息
async function sendMessage(message, context) {
    const response = await fetch('/api/manus/send_message', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            message: message,
            context: context
        })
    });
    return await response.json();
}

// 使用示例
startManusService().then(result => {
    console.log('服務啟動結果:', result);

    return sendMessage('測試消息', {
        repository: 'communitypowerauto'
    });
}).then(result => {
    console.log('消息發送結果:', result);
});
```

高級功能

1. 任務批量處理

系統支持批量處理多個任務，自動遍歷任務列表並執行指定操作。

```python
async def process_all_tasks():
    controller = ManusAdvancedController()
    await controller.start()
    await controller.login()

    # 獲取所有任務
    tasks = await controller.get_task_list()

    for task in tasks:
        print(f"處理任務: {task.title}")

        # 獲取對話歷史
        messages = await controller.get_conversation_history(task)

        # 下載文件
        files = await controller.batch_download_files()

        # 分析對話
        # ... 執行分析邏輯
```

2. 文件智能分類

系統能夠根據文件內容和類型進行智能分類，並創建相應的目錄結構。

分類邏輯: - 代碼文件：根據擴展名識別編程語言 - 文檔文件：根據內容識別文檔類型 - 圖片文件：根據格式和用途分類 - 鏈接：根據域名和內容分類

3. 對話內容分析

系統提供深度的對話內容分析功能，包括：

- **情感分析**: 識別對話中的情感傾向
- **主題提取**: 自動提取對話主題
- **關鍵詞識別**: 識別重要的技術關鍵詞
- **複雜度評估**: 評估對話的技術複雜度

4. 智能介入策略

系統實現了多層次的智能介入策略：

介入觸發條件: - 用戶提出技術問題 - 對話出現錯誤或困惑 - 需要專業建議的場景 - 複雜度超過閾值的討論

介入類型: - **建議型介入**: 提供技術建議和最佳實踐 - **糾錯型介入**: 指出錯誤並提供正確方案 - **補充型介入**: 補充遺漏的重要信息 - **引導型介入**: 引導對話朝正確方向發展

監控和日誌

日誌系統

系統提供完整的日誌記錄功能，包括：

- **操作日誌**: 記錄所有用戶操作
- **錯誤日誌**: 記錄系統錯誤和異常
- **性能日誌**: 記錄響應時間和性能指標
- **安全日誌**: 記錄登錄和安全相關事件

監控指標

- **服務可用性**: 監控服務運行狀態
- **響應時間**: 監控API響應時間
- **成功率**: 監控操作成功率
- **資源使用**: 監控CPU和內存使用

安全考慮

1. 認證和授權

- 使用安全的登錄憑證存儲
- 實現API密鑰認證機制
- 支持角色基礎的訪問控制

2. 數據安全

- 敏感數據加密存儲
- 安全的文件下載機制
- 用戶數據隱私保護

3. 網絡安全

- HTTPS加密傳輸
- 跨域請求安全控制
- 防止XSS和CSRF攻擊

故障排除

常見問題

1. 瀏覽器啟動失敗

症狀: Playwright無法啟動Chromium **解決方案**:

```
 playwright install chromium
export PLAYWRIGHT_BROWSERS_PATH=/home/ubuntu/.cache/ms-playwright
```

2. 登錄失敗

症狀: 無法自動登錄Manus **解決方案**: - 檢查登錄憑證是否正確 - 確認Manus網站可訪問 - 檢查網絡連接

3. 文件下載失敗

症狀: 無法下載文件或下載不完整 **解決方案**: - 檢查下載目錄權限 - 確認文件URL有效 - 增加下載超時時間

調試模式

啟用調試模式以獲取詳細日誌：

```
 # 設置日誌級別
logging.basicConfig(level=logging.DEBUG)

# 啟用Playwright調試
config = {
    'headless': False,  # 顯示瀏覽器
    'slow_mo': 2000,    # 增加操作間隔
}
```

性能優化

1. 瀏覽器優化

- 使用無頭模式減少資源消耗
- 禁用不必要的瀏覽器功能
- 優化頁面載入等待時間

2. 並發處理

- 實現異步操作
- 使用連接池管理
- 批量處理優化

3. 緩存策略

- 實現智能緩存機制
- 緩存常用數據
- 減少重複操作

擴展和定制

1. 自定義選擇器

系統支持自定義CSS選擇器以適應不同的頁面結構：

```
custom_selectors = {
    'task_list': '.custom-task-list',
    'message_input': '.custom-input',
    'send_button': '.custom-send-btn'
}
```

2. 插件系統

支持插件擴展功能：

```
class CustomPlugin:
    async def process_message(self, message):
        # 自定義消息處理邏輯
        return processed_message

    async def analyze_conversation(self, messages):
        # 自定義對話分析邏輯
        return analysis_result
```

3. 配置文件

支持外部配置文件：

```
{
  "manus": {
    "url": "https://manus.chat",
    "credentials": {
      "email": "user@example.com",
      "password": "password"
    }
  },
  "browser": {
    "headless": true,
    "timeout": 60000
  },
  "download": {
    "directory": "/tmp/downloads",
    "categories": ["Documents", "Images", "Code files"]
  }
}
```

版本歷史

v1.0.0 (2025-06-22)

- 初始版本發布
- 實現基礎Manus自動化功能
- 支持任務列表遍歷
- 實現文件分類下載
- 提供完整API接口

未來計劃

- **v1.1.0**: 增加更多文件類型支持
- **v1.2.0**: 實現智能對話摘要
- **v1.3.0**: 添加多用戶支持
- **v2.0.0**: 完整的AI集成和自動化工作流

結論

PowerAutomation Playwright集成系統提供了完整的Manus頁面自動化解決方案。通過先進的瀏覽器自動化技術，系統能夠高效地處理任務列表遍歷、文件管理、對話歷史獲取和智能分析等複雜任務。

系統的模塊化設計和豐富的API接口使其易於集成到現有的工作流程中，為用戶提供了強大的自動化能力和智能輔助功能。

隨著技術的不斷發展，系統將持續優化和擴展，為用戶提供更加智能和高效的自動化體驗。