虚拟机运行机制及类关系文档

---

1. 整体架构概述

该虚拟机是一个基于指令集的自动化任务引擎，通过加载并执行特定格式的 JSON 指令序列，支持变量管理、流程控制（如循环）及日志记录。其核心设计目标包括：

- **模块化**：通过抽象指令基类实现可扩展性。
- **状态隔离**：通过执行上下文（`ExecutionContext`）管理运行时状态。
- **多场景支持**：已实现基础指令，可扩展至 SAP、Web 自动化等场景。

---

2. 核心类及其职责

2.1 `VirtualMachine` 类

- **职责：**
  - 加载并解析 JSON 指令文件。
  - 调度指令执行，维护全局执行流程。
  - 集成日志系统（Serilog），记录运行时状态。
- **关键方法：**
  - `LoadAndExecute(string jsonPath)`：入口方法，加载 JSON 并执行指令。
  - `ParseInstructions(JArray jArray)`：递归解析 JSON 指令树。
  - `ExecuteCore(List<Instruction> instructions)`：执行指令序列，处理异常。

2.2 `ExecutionContext` 类

- **职责：**
  - 存储运行时状态：变量表（`Variables`）、循环堆栈（`LoopStack`）。
  - 提供动态属性访问（如 `CurrentLoopIndex`）。
  - 绑定日志接口（`ILogger`），支持指令级日志输出。
- **关键属性：**
  - `Variables`：字典，存储用户定义的变量。
  - `LoopStack`：堆栈，记录嵌套循环的当前索引。
  - `CurrentLoopIndex`：动态属性，返回当前循环索引。

2.3 `Instruction` 基类及派生类

- **基类：**

```
public abstract class Instruction {
    public abstract void Execute(ExecutionContext context);
}
```

- **派生类：**

1. `AssignInstruction`
   - **功能**：变量赋值。
   - **JSON 示例**：

```
{ "Type": "Assign", "Variable": "counter", "Value": 5 }
```

2. `PrintInstruction`
   - **功能**：打印变量或上下文属性。
   - **JSON 示例**：

```
{ "Type": "Print", "Variable": "ExecutionContext.CurrentLoopIndex"
}
```

3. `LoopInstruction`
   - **功能**：循环执行子指令。
   - **JSON 示例**：

```
{
  "Type": "Loop",
  "VariableIterations": "maxCount",
  "Instructions": [ ... ]
}
```

---

3. 类关系与交互

```
classDiagram
    class VirtualMachine {
        -ExecutionContext _context
        -ILogger _logger
        +LoadAndExecute()
        -ParseInstructions()
        -ExecuteCore()
    }

    class ExecutionContext {
        +Dictionary Variables
        +Stack LoopStack
        +ILogger Logger
        +PushLoopIndex()
        +PopLoopIndex()
        +int CurrentLoopIndex
    }

    class Instruction {
```

```
        <>
        +Execute(ExecutionContext context)
    }

    class AssignInstruction {
        +string Variable
        +JToken Value
        +Execute()
    }

    class PrintInstruction {
        +string Variable
        +Execute()
    }

    class LoopInstruction {
        +int? FixedIterations
        +string VariableIterations
        +List Instructions
        +Execute()
    }

    VirtualMachine --> ExecutionContext : 维护运行时状态
    VirtualMachine --> Instruction : 解析并调度执行
    AssignInstruction --|> Instruction
    PrintInstruction --|> Instruction
    LoopInstruction --|> Instruction
    LoopInstruction --> Instruction : 包含子指令
    ExecutionContext --> ILogger : 日志输出
```

---

4. 指令执行流程

1. **加载与解析**：
   - `VirtualMachine` 读取 JSON 文件，递归解析为 `Instruction` 对象树。
   - 示例：循环指令解析时，其子指令会被递归转换为 `PrintInstruction` 或嵌套的 `LoopInstruction`。
2. **执行阶段**：
   - **变量赋值**：`AssignInstruction` 更新 `Variables` 字典。
   - **循环控制**：`LoopInstruction` 根据迭代次数（固定值或变量值）执行子指令，维护 `LoopStack`。
   - **状态传递**：所有指令通过 `ExecutionContext` 共享变量和循环索引。
3. **日志记录**：
   - 关键操作（如变量修改、循环迭代）通过 `ILogger` 输出结构化日志。
   - 示例：`PrintInstruction` 输出 `CurrentLoopIndex` 时，直接访问 `ExecutionContext` 属性。

---

5. 扩展性设计

1. **新增指令类型：**
    - 继承 `Instruction` 基类，实现 `Execute` 方法。
    - 更新 `ParseInstruction` 方法，支持新指令类型的 JSON 解析。
    - 示例：添加 `SAPClickInstruction` 以支持 SAP 控件操作。
2. **集成外部系统：**
    - 在指令中调用外部库（如 SAP GUI Scripting API、Playwright）。
    - 通过 `ExecutionContext` 维护会话状态（如 SAP 连接句柄）。
3. **增强流程控制：**
    - 实现 `IfElseInstruction` 支持条件分支。
    - 添加 `TryCatchInstruction` 实现错误恢复机制。

---

6. 错误处理与调试

- **异常捕获：**
    - `VirtualMachine` 在 `ExecuteCore` 中捕获全局异常，记录错误堆栈。
    - 指令内部可抛出业务异常（如 `KeyNotFoundException`）。
- **日志分析：**
    - 通过 Serilog 输出到控制台和文件，支持过滤关键事件（如未定义变量警告）。
    - 示例：循环未执行时，检查子指令是否成功解析。

---

7. 总结

该虚拟机通过模块化设计和状态隔离机制，实现了灵活的任务自动化能力。核心类职责明确，指令扩展简单，日志系统为调试提供支持。未来可通过新增指令类型和集成外部 API，进一步扩展应用场景。