Sequential agents - Agent Development Kit

Source URL: https://google.github.io/adk-docs/agents/workflow-agents/sequential-agents/

Sequential agents¶

The `SequentialAgent`¶

The `SequentialAgent` is a workflow agent that executes its sub-agents in the order they are specified in the list.

Use the `SequentialAgent` when you want the execution to occur in a fixed, strict order.

Example¶

- You want to build an agent that can summarize any webpage, using two tools: `Get Page Contents` and `Summarize Page`. Because the agent must always call `Get Page Contents` before calling `Summarize Page` (you can't summarize from nothing!), you should build your agent using a `SequentialAgent`.

As with other workflow agents, the `SequentialAgent` is not powered by an LLM, and is thus deterministic in how it executes. That being said, workflow agents are concerned only with their execution (i.e. in sequence), and not their internal logic; the tools or sub-agents of a workflow agent may or may not utilize LLMs.

How it works¶

When the `SequentialAgent`'s `Run Async` method is called, it performs the following actions:

1. **Iteration:** It iterates through the sub agents list in the order they were provided.

2. **Sub-Agent Execution:** For each sub-agent in the list, it calls the sub-agent's `Run Async` method.

Sequential Agent

Full Example: Code Development Pipeline¶

Consider a simplified code development pipeline:

- **Code Writer Agent:** An LLM Agent that generates initial code based on a specification.
- **Code Reviewer Agent:** An LLM Agent that reviews the generated code for errors, style issues, and adherence to best practices. It receives the output of the Code Writer Agent.
- **Code Refactorer Agent:** An LLM Agent that takes the reviewed code (and the reviewer's comments) and refactors it to improve quality and address issues.

A `SequentialAgent` is perfect for this:

```
SequentialAgent(sub_agents=[CodeWriterAgent, CodeReviewerAgent, CodeR
```

This ensures the code is written, *then* reviewed, and *finally* refactored, in a strict, dependable order. **The output from each sub-agent is passed to the next by storing them in state via Output Key**.

Code

PythonJava

```
# Part of agent.py --> Follow https://google.github.io/adk-docs/get-s

# --- 1. Define Sub-Agents for Each Pipeline Stage ---

# Code Writer Agent
# Takes the initial specification (from user query) and writes code.
code_writer_agent = LlmAgent(
    name="CodeWriterAgent",
```

```
    model=GEMINI_MODEL,
    # Change 3: Improved instruction
    instruction="""You are a Python Code Generator.
Based *only* on the user's request, write Python code that fulfills th
Output *only* the complete Python code block, enclosed in triple backt
Do not add any other text before or after the code block.
""",
    description="Writes initial Python code based on a specification."
    output_key="generated_code" # Stores output in state['generated_co
)

# Code Reviewer Agent
# Takes the code generated by the previous agent (read from state) and
code_reviewer_agent = LlmAgent(
    name="CodeReviewerAgent",
    model=GEMINI_MODEL,
    # Change 3: Improved instruction, correctly using state key inject
    instruction="""You are an expert Python Code Reviewer.
    Your task is to provide constructive feedback on the provided code

    **Code to Review:**
    ```python
    {generated_code}
    ```

Review Criteria:
1. **Correctness:** Does the code work as intended? Are there logic e
2. **Readability:** Is the code clear and easy to understand? Follows
3. **Efficiency:** Is the code reasonably efficient? Any obvious perf
4. **Edge Cases:** Does the code handle potential edge cases or inval
5. **Best Practices:** Does the code follow common Python best practi

Output:
Provide your feedback as a concise, bulleted list. Focus on the most i
If the code is excellent and requires no changes, simply state: "No ma
Output *only* the review comments or the "No major issues" statement.
```

```
""",
    description="Reviews code and provides feedback.",
    output_key="review_comments", # Stores output in state['review_com
)

# Code Refactorer Agent
# Takes the original code and the review comments (read from state) an
code_refactorer_agent = LlmAgent(
    name="CodeRefactorerAgent",
    model=GEMINI_MODEL,
    # Change 3: Improved instruction, correctly using state key inject
    instruction="""You are a Python Code Refactoring AI.
Your goal is to improve the given Python code based on the provided re

  **Original Code:**
  ```python
  {generated_code}
  ```

  Review Comments:
  {review_comments}

Task:
Carefully apply the suggestions from the review comments to refactor t
If the review comments state "No major issues found," return the origi
Ensure the final code is complete, functional, and includes necessary

Output:
Output *only* the final, refactored Python code block, enclosed in tri
Do not add any other text before or after the code block.
""",
    description="Refactors code based on review comments.",
    output_key="refactored_code", # Stores output in state['refactored
)

--- 2. Create the SequentialAgent ---
```

```
# This agent orchestrates the pipeline by running the sub_agents in or
code_pipeline_agent = SequentialAgent(
    name="CodePipelineAgent",
    sub_agents=[code_writer_agent, code_reviewer_agent, code_refactore
    description="Executes a sequence of code writing, reviewing, and r
    # The agents will run in the order provided: Writer -> Reviewer ->
)

# For ADK tools compatibility, the root agent must be named `root_ager
root_agent = code_pipeline_agent
```

```
 import com.google.adk.agents.LlmAgent;
import com.google.adk.agents.SequentialAgent;
import com.google.adk.events.Event;
import com.google.adk.runner.InMemoryRunner;
import com.google.adk.sessions.Session;
import com.google.genai.types.Content;
import com.google.genai.types.Part;
import io.reactivex.rxjava3.core.Flowable;

public class SequentialAgentExample {

  private static final String APP_NAME = "CodePipelineAgent";
  private static final String USER_ID = "test_user_456";
  private static final String MODEL_NAME = "gemini-2.0-flash";

  public static void main(String[] args) {
    SequentialAgentExample sequentialAgentExample = new SequentialAgen
    sequentialAgentExample.runAgent(
        "Write a Java function to calculate the factorial of a number.
  }

  public void runAgent(String prompt) {
```

```
    LlmAgent codeWriterAgent =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("CodeWriterAgent")
            .description("Writes initial Java code based on a specific
            .instruction(
                """
                You are a Java Code Generator.
                Based *only* on the user's request, write Java code th
                Output *only* the complete Java code block, enclosed i
                Do not add any other text before or after the code blo
                """)
            .outputKey("generated_code")
            .build();

    LlmAgent codeReviewerAgent =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("CodeReviewerAgent")
            .description("Reviews code and provides feedback.")
            .instruction(
                """
                    You are an expert Java Code Reviewer.
                    Your task is to provide constructive feedback on t

                    **Code to Review:**
                    ```java
                    {generated_code}
                    ```

                    Review Criteria:
                    1. **Correctness:** Does the code work as intende
                    2. **Readability:** Is the code clear and easy to
                    3. **Efficiency:** Is the code reasonably efficie
                    4. **Edge Cases:** Does the code handle potential
                    5. **Best Practices:** Does the code follow commo
```

```
                    **Output:**
                    Provide your feedback as a concise, bulleted list.
                    If the code is excellent and requires no changes,
                    Output *only* the review comments or the "No major
                """)
            .outputKey("review_comments")
            .build();

    LlmAgent codeRefactorerAgent =
        LlmAgent.builder()
            .model(MODEL_NAME)
            .name("CodeRefactorerAgent")
            .description("Refactors code based on review comments.")
            .instruction(
                """
                You are a Java Code Refactoring AI.
                Your goal is to improve the given Java code based on t

                  **Original Code:**
                  ```java
                  {generated_code}
                  ```

                  Review Comments:
                  {review_comments}

                Task:
                Carefully apply the suggestions from the review commen
                If the review comments state "No major issues found,"
                Ensure the final code is complete, functional, and inc

                Output:
                Output *only* the final, refactored Java code block, e
                Do not add any other text before or after the code blo
                """)
```

```
            .outputKey("refactored_code")
            .build();

    SequentialAgent codePipelineAgent =
        SequentialAgent.builder()
            .name(APP_NAME)
            .description("Executes a sequence of code writing, reviewi
            // The agents will run in the order provided: Writer -> Re
            .subAgents(codeWriterAgent, codeReviewerAgent, codeRefacto
            .build();

    // Create an InMemoryRunner
    InMemoryRunner runner = new InMemoryRunner(codePipelineAgent, APP_
    // InMemoryRunner automatically creates a session service. Create
    Session session = runner.sessionService().createSession(APP_NAME,
    Content userMessage = Content.fromParts(Part.fromText(prompt));

    // Run the agent
    Flowable<Event> eventStream = runner.runAsync(USER_ID, session.id(

    // Stream event response
    eventStream.blockingForEach(
        event -> {
          if (event.finalResponse()) {
            System.out.println(event.stringifyContent());
          }
        });
  }
}
```