FAQ & Troubleshooting

Sofia Billger Bergström

FAQ & Troubleshooting

1. How does the agent remember conversations?

The agent uses a local vector store (**ChromaDB**) to save the history of your interactions. A directory named **.code_agent_memory** is created in your project root to store this data. This allows the agent to use past conversations as context for new requests, improving its relevance and accuracy over time.

2. Why does the agent run locally?

Privacy, Security, and Cost. By running the LLM and all tools on your local machine, your code and data never leave your computer. This is crucial for maintaining the confidentiality of proprietary or sensitive projects. It also means there are no API costs associated with using the agent.

3. Can I use a different local LLM?

Yes. The agent is configured via the **code_agent/config/llm_config.json** file. You can change the **ollama_model** to any other model you have available in Ollama (e.g., **llama3**, **codellama**, etc.). The **create_llm** function is designed to be easily adaptable to other local LLM providers as well.

4. What happens if the Ollama server is not running?

If the agent cannot connect to the Ollama server, the **create_llm** function will automatically create a **fallback LLM** . This fallback model doesn't perform any actions but returns a clear error message, ensuring that the application remains responsive and informs you of the connection issue.

5. Is it safe to let the agent edit my files?

The agent's file system tools are designed with safety in mind: - **Sandboxed:** All file operations are restricted to the project's root directory. The tools will prevent any attempt to access files outside of this directory. - **Backups:** The **`edit-file`** tool automatically creates a backup (**`.bak`**) of any file it modifies, allowing you to easily revert any unwanted changes.

For maximum security, especially on critical projects, consider running the agent in a containerized environment like Docker.

6. How do I add a new tool to the agent?

1. Create a new Python file in the **`code_agent/tools/`** directory.
2. Define a new class that inherits from **`langchain_core.tools.BaseTool`**.
3. Implement the **`_run`** method with your tool's logic.
4. Add your new tool to the **`create_default_tools`** function in **`code_agent/agents/base_agent.py`**.

See the existing tools for examples.