MCP DeepWiki Server Usage Examples

This document provides practical examples of how to use the MCP DeepWiki Server with different AI assistants and clients.

Example Tool Calls

Basic Repository Fetch

```
{
  "name": "deepwiki_fetch",
  "arguments": {
    "url": "https://deepwiki.com/modelcontextprotocol/typescript-sdk",
    "mode": "aggregate",
    "maxDepth": 5
  }
}
```

Shorthand Repository Format

```
{
  "name": "deepwiki_fetch",
  "arguments": {
    "url": "neka-nat/freecad-mcp",
    "mode": "pages",
    "maxDepth": 3
  }
}
```

Large Repository with Depth Limit

```
{
  "name": "deepwiki_fetch",
  "arguments": {
    "url": "openai/openai-agents-python",
    "mode": "aggregate",
    "maxDepth": 2
  }
}
```

Conversation Examples

Claude Desktop Example

User: “How does the MCP TypeScript SDK handle tool registration?”

Claude: I’ll fetch the documentation for the MCP TypeScript SDK to give you a comprehensive answer.

[Claude calls deepwiki_fetch with the MCP TypeScript SDK repository]

Response: Based on the MCP TypeScript SDK documentation, tool registration is handled through the `registerTool` method on the `McpServer` instance. Here's how it works:

1. **Tool Definition**: Tools are registered with a name, description, and input schema
2. **Input Validation**: The SDK uses Zod schemas for input validation
3. **Handler Function**: Each tool has an async handler that processes the input
4. **Response Format**: Tools return structured content arrays

[Detailed explanation with code examples from the fetched documentation]

Cursor Example

User: "Explain the architecture of the FreeCAD MCP server and how it integrates with FreeCAD"

[Cursor automatically detects the need for repository information and calls deepwiki_fetch]

Response: The FreeCAD MCP server provides a bridge between MCP clients and FreeCAD's Python API. Here's the architecture:

[Detailed explanation based on fetched documentation]

VS Code Copilot Example

User: "What are the main classes in the OpenAI agents Python library?"

[VS Code Copilot calls deepwiki_fetch to get the repository documentation]

Response: Based on the OpenAI agents Python library documentation, the main classes include:

[Structured overview of classes and their purposes]

Integration Scenarios

Local Development Setup

1. **Claude Desktop for Code Review**
   ```
   bash
   # User asks Claude to review repository architecture
   "Please review the architecture of the react-router repository and suggest improvements"
   ```

2. **Cursor for Feature Implementation**
   ```
   bash
   # User asks Cursor to implement a feature based on existing patterns
   "Implement a new tool following the patterns used in the MCP TypeScript SDK"
   ```

3. **VS Code for Documentation**
   ```
   bash
   # User asks Copilot to explain complex code patterns
   "Explain how error handling works in this codebase"
   ```

Remote Server Setup

For team environments, run the server in HTTP mode:

```
# Start server
PORT=4000 node dist/index.js

# Team members can connect their MCP clients to:
# http://your-server:4000/mcp
```

Advanced Usage Patterns

Comparative Analysis

```
{
  "workflow": "compare_repositories",
  "steps": [
    {
      "tool": "deepwiki_fetch",
      "args": {
        "url": "modelcontextprotocol/typescript-sdk",
        "mode": "aggregate"
      }
    },
    {
      "tool": "deepwiki_fetch",
      "args": {
        "url": "modelcontextprotocol/python-sdk",
        "mode": "aggregate"
      }
    }
  ],
  "prompt": "Compare the architecture and API design between the TypeScript and Python
MCP SDKs"
}
```

Deep Dive Investigation

```
{
  "tool": "deepwiki_fetch",
  "args": {
    "url": "large-repository/complex-project",
    "mode": "pages",
    "maxDepth": 8
  },
  "follow_up": "Analyze the modular structure and identify the core components"
}
```

Documentation Generation

```
{
  "workflow": "generate_docs",
  "steps": [
    {
      "tool": "deepwiki_fetch",
      "args": {
        "url": "your-org/your-project",
        "mode": "aggregate"
      }
    }
  ],
  "prompt": "Generate a comprehensive README based on the existing documentation struc-
ture"
}
```

Error Handling Examples

Invalid Repository

```
{
  "tool": "deepwiki_fetch",
  "args": {
    "url": "nonexistent/repository"
  }
}
```

Response:

```
Error fetching DeepWiki content: Repository not found or not accessible
```

Security Violation

```
{
  "tool": "deepwiki_fetch",
  "args": {
    "url": "https://malicious-site.com/repo"
  }
}
```

Response:

```
Error fetching DeepWiki content: Domain not allowed. Only deepwiki.com are permitted.
```

Network Issues

```
{
  "tool": "deepwiki_fetch",
  "args": {
    "url": "timeout-repository/large-repo",
    "maxDepth": 50
  }
}
```

Response:

```
Error fetching DeepWiki content: Request timeout - try reducing maxDepth or check
network connectivity
```

Best Practices

Optimizing Performance

1. **Use Appropriate Depth**: Start with low `maxDepth` (2-3) for large repositories
2. **Choose Right Mode**: Use “aggregate” for comprehensive analysis, “pages” for structured exploration
3. **Cache Results**: Some clients may cache tool results automatically

Effective Prompting

1. **Be Specific**: “Explain the authentication flow in the OAuth library” vs “Tell me about this repository”
2. **Reference Components**: “How does the DatabaseManager class handle connections?”
3. **Compare Patterns**: “Compare the error handling in this repository with industry standards”

Security Considerations

1. **Trusted Sources**: Only use the tool with repositories you trust
2. **Validate URLs**: The server validates URLs, but double-check your inputs
3. **Rate Limiting**: Be mindful of making too many requests in succession

Troubleshooting

Common Issues and Solutions

1. **Tool Not Available**
   - Check MCP server is running and connected
   - Verify configuration file syntax
   - Restart your MCP client

2. **Empty Results**
   - Repository might not be available on DeepWiki
   - Try the direct DeepWiki URL in a browser first
   - Check repository name spelling

3. **Partial Content**
   - Increase `maxDepth` parameter

   - Repository might have unusual structure
   - Some pages might be temporarily unavailable

4. **Performance Issues**
   - Reduce `maxDepth` for large repositories
   - Use “aggregate” mode for faster processing
   - Check network connectivity

Support

For additional help:
- Check the main README.md for setup instructions
- Review server logs for detailed error messages
- Test with the provided example repositories first