PolarVista / **Xcode-mcp-server** Public

Code | Issues 3 | Pull requests | Actions | Projects | Security and quality

1 Branch 0 Tags Go to file Go to file Code

PolarVista Implement warning filtering 751adea · 2 years ago

build	Implement warning filtering	2 years ago
node_modules	Initial commit, implements b…	2 years ago
src	Implement warning filtering	2 years ago
.DS_Store	Initial commit, implements b…	2 years ago
README.md	Fix repo URL	2 years ago
package-lock.json	Initial commit, implements b…	2 years ago
package.json	Initial commit, implements b…	2 years ago
tsconfig.json	Initial commit, implements b…	2 years ago

README

Xcode MCP Server

A Model Context Protocol server for building Xcode projects directly from LLM applications

The Xcode MCP Server provides a Model Context Protocol interface for building and testing Xcode projects. It enables AI assistants to directly trigger builds, run tests, monitor progress, and access logs through a standardized interface.

Features

- Build Xcode projects with custom schemes and configurations

- Run unit tests with granular control (run specific tests or skip tests)
- Generate code coverage reports
- Stream build and test logs in real-time
- Access detailed build and test reports
- JSON-formatted output
- Automatic log persistence

Requirements

- Node.js 16+
- Xcode Command Line Tools
- TypeScript
- MCP-compatible client (e.g., Claude Desktop)

Installation

```
# Clone the repository
git clone https://github.com/PolarVista/Xcode-mcp-server.git
cd xcode-mcp-server

# Install dependencies
npm install

# Build the server
npm run build
```

Usage with Claude Desktop

1. Start the server:

   ```
   npm run start /path/to/build/logs/directory
   ```

2. In Claude Desktop settings:

   ```
         "command": "node",
         "args": ["/path/to/the/xcode-mcp-server/build/index.js",
                  "/path/to/your/project/folder"],
         "env": {
   ```

```
            "PATH": "/usr/bin:/bin:/usr/local/bin:/usr/sbin:/sbin"
        }
```

Available Tools

build_project

Builds an Xcode project with specified parameters.

Parameters:

- `projectPath` (required): Path to the .xcodeproj or .xcworkspace
- `scheme` (required): Build scheme name
- `configuration` (optional): Build configuration (Debug/Release, defaults to Debug)
- `destination` (optional): Build destination (defaults to "platform=iOS Simulator,name=iPhone 15 Pro")

Example usage in Claude:

```
build_project({
  projectPath: "/path/to/Project.xcodeproj",
  scheme: "MyApp",
  configuration: "Debug"
})
```

run_tests

Runs unit tests with optional filtering.

Parameters:

- `projectPath` (required): Path to the .xcodeproj or .xcworkspace
- `scheme` (required): Test scheme name
- `testIdentifier` (optional): Specific test to run (e.g., 'MyTests/testExample')
- `skipTests` (optional): Array of test identifiers to skip
- `configuration` (optional): Build configuration (Debug/Release, defaults to Debug)
- `destination` (optional): Test destination (defaults to "platform=iOS Simulator,name=iPhone 15 Pro")

Example usage in Claude:

```
run_tests({
  projectPath: "/path/to/Project.xcodeproj",
  scheme: "MyAppTests",
  testIdentifier: "LoginTests/testSuccessfulLogin",
  skipTests: ["PerformanceTests/testLargeDataLoad"],
  configuration: "Debug"
})
```

Logs

- All logs are stored in the specified base directory under `build-logs/`
- Build operations create:
  - Plain text log (`build-[timestamp].log`)
  - JSON-formatted log (`build-[timestamp].log.json`)
  - Xcode report (`report-[timestamp].txt`)
- Test operations create:
  - Test log (`test-[timestamp].log`)
  - JSON-formatted log (`test-[timestamp].log.json`)
  - Test report (`test-report-[timestamp].txt`)
  - Code coverage report (`coverage-[timestamp].txt`)

Latest log (build or test) is accessible via the [illegible] resource

Releases

No releases published

Packages

No packages published

Contributors 1

PolarVista

Languages


TypeScript 50.1%
JavaScript 49.9%