readme.md for @push.rocks/smartexit

A library for managing graceful shutdowns of Node.js processes by handling cleanup operations, including terminating child processes.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Install

To install `@push.rocks/smartexit`, use pnpm:

```
pnpm install @push.rocks/smartexit
```

Usage

`@push.rocks/smartexit` helps you gracefully shut down Node.js applications by running cleanup operations before the process exits. It handles signal interception (`SIGINT`, `SIGTERM`), uncaught exception management, and child process termination.

Basic Setup

```
import { SmartExit } from '@push.rocks/smartexit';
```

```
// Create an instance - this automatically hooks into process signals
const smartExit = new SmartExit();
```

Registering Cleanup Functions

Define custom async cleanup functions that execute before the process exits:

```
smartExit.addCleanupFunction(async () => {
  console.log('Closing database connections...');
  await database.close();
});

smartExit.addCleanupFunction(async () => {
  console.log('Flushing logs...');
  await logger.flush();
});
```

Managing Child Processes

Track spawned child processes so they get properly terminated on exit:

```
import { spawn } from 'child_process';

const childProcess = spawn('node', ['worker.js']);
smartExit.addProcess(childProcess);

// Remove from tracking if the process ends naturally
childProcess.on('exit', () => {
  smartExit.removeProcess(childProcess);
});
```

Kill Process Trees by PID

Terminate an entire process tree using the static `killTreeByPid` method:

```
import { SmartExit, type TProcessSignal } from '@push.rocks/smartexit';
```

```
// Kill with default SIGKILL
await SmartExit.killTreeByPid(12345);

// Kill with a specific signal
await SmartExit.killTreeByPid(12345, 'SIGTERM');
```

Triggering Cleanup Manually

While `SmartExit` automatically hooks into `SIGINT`, `SIGTERM`, and uncaught exceptions, you can manually trigger cleanup:

```
await smartExit.killAll();
process.exit(0);
```

Integrating with Express

Gracefully close an Express server on shutdown:

```
import express from 'express';
import { SmartExit } from '@push.rocks/smartexit';

const app = express();
const smartExit = new SmartExit();

const server = app.listen(3000, () => {
  console.log('Server running on port 3000');
});

smartExit.addCleanupFunction(async () => {
  console.log('Closing Express server...');
  await new Promise<void>((resolve) => server.close(() => resolve()));
});
```

Available Process Signals

The `TProcessSignal` type provides all standard POSIX signals:

```
import type { TProcessSignal } from '@push.rocks/smartexit';

// Examples: 'SIGINT', 'SIGTERM', 'SIGKILL', 'SIGHUP', 'SIGUSR1', 'SIGUSR2', etc.
```

How It Works

When you create a `SmartExit` instance, it automatically:

1. **Hooks `SIGINT`** (Ctrl+C): Runs cleanup and exits with code 0
2. **Hooks process `exit`**: Runs cleanup when `process.exit(0)` is called
3. **Catches uncaught exceptions**: Logs the error, runs cleanup, and exits with code 1

On `killAll()`, it:

- Sends `SIGINT` to all tracked child processes
- Waits 10 seconds, then sends `SIGKILL` if processes are still alive
- Executes all registered cleanup functions

License and Legal Information

Trademarks

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:10:36 UTC by foss.global Team
Updated 2026-03-28 12:17:24 UTC by foss.global Team