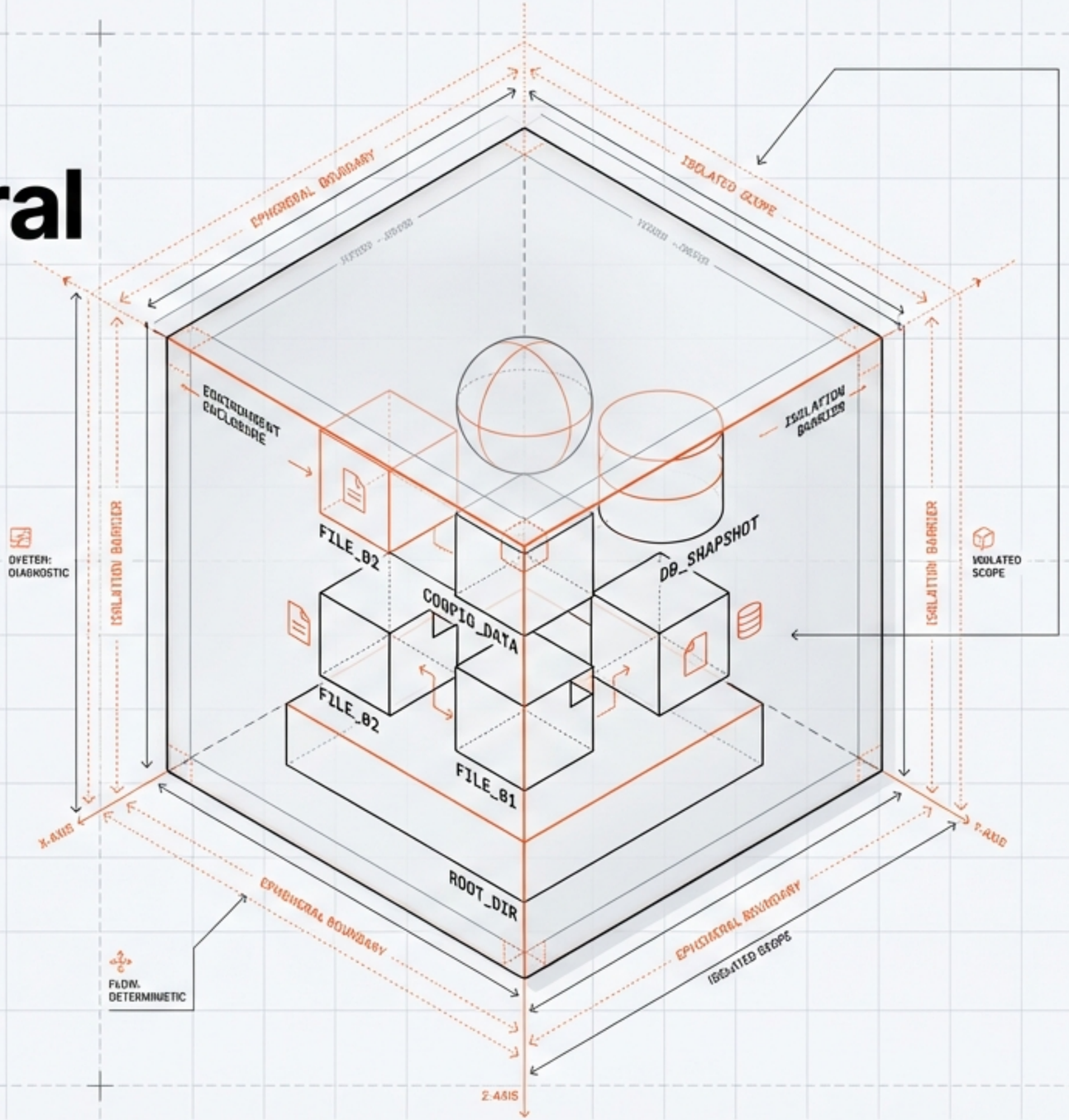
Temp Utilities: The Art of Ephemeral Testing

Deterministic resource management for the modern developer.



JetBrains Mono

```
PACKAGE: osbot_utils.testing.*
VERSION: v3.69.2
STATUS:  STABLE
```

The Friction of Manual Resource Management

Testing requires resources. Managing them manually is the enemy of reliability.

Leftover Artifacts: Files lingering after a crash.

Port Conflicts: 'Address already in use' errors errors on CI/CD.

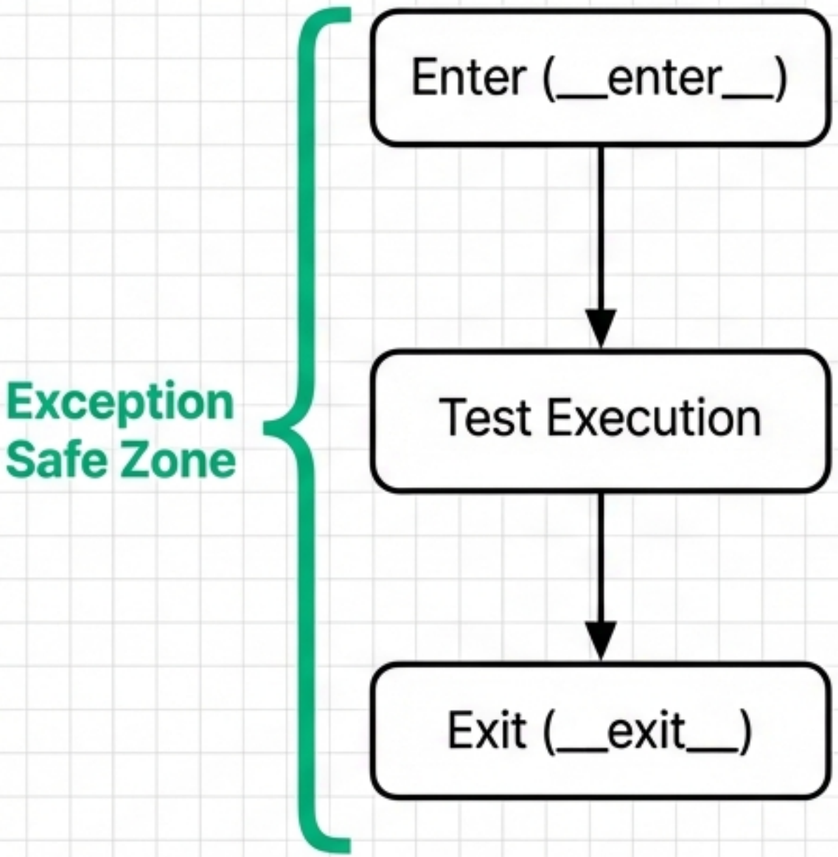
legacy_test.py

```python
def test_legacy_workflow():
    path = '/tmp/test_file.txt'
    try:
        f = open(path, 'w')
        f.write('data')
        # ... tedious logic ...
    finally:
        # The Boilerplate Trap
        if os.path.exists(path):
            os.remove(path)
```

Boilerplate Trap: Writing 10 lines of setup for 1 line of assertion.

The Context Manager Paradigm

Enter creates. Exit cleans.




```
with Temp_File(contents='data') as temp_file:
    assert temp_file.contents() == 'data'

# Context exits: File is automatically deleted.
```

The `with` Statement
Resources wrapped in context managers.

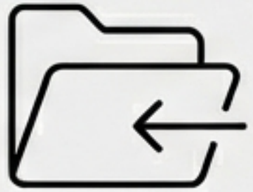
Automatic Cleanup
Deletion happens even if exceptions occur.

Zero-Ceremony
No manual teardown logic required.

The Ephemeral Ecosystem

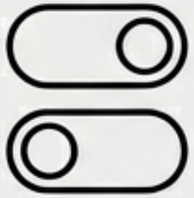
A complete toolkit for isolated testing.

Filesystem



`Temp_File`
`Temp_Folder`

Environment



`Temp_Env_Vars`
`Temp_Sys_Path`

Network

`Temp_Web_Server`

Archives

`Temp_Zip`
`Temp_Zip_In_Memory`

Design Principles: Test-First Mindset • Chainable & Nestable

Files & Folders: Precision on Disk

Temp_File

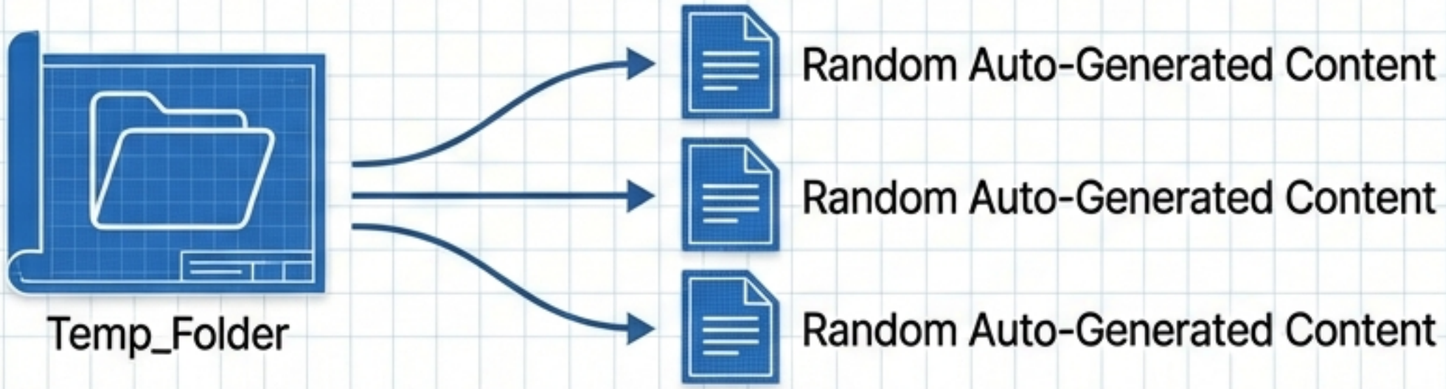
```
Temp_File(contents='...', extension='tmp')
```

- **Rich Defaults:** Auto-generates random content if empty.
- **Key Methods:** `path()`, `contents()`, `write()`
- **Pro Tip:** `return_file_path=True` returns string path directly.

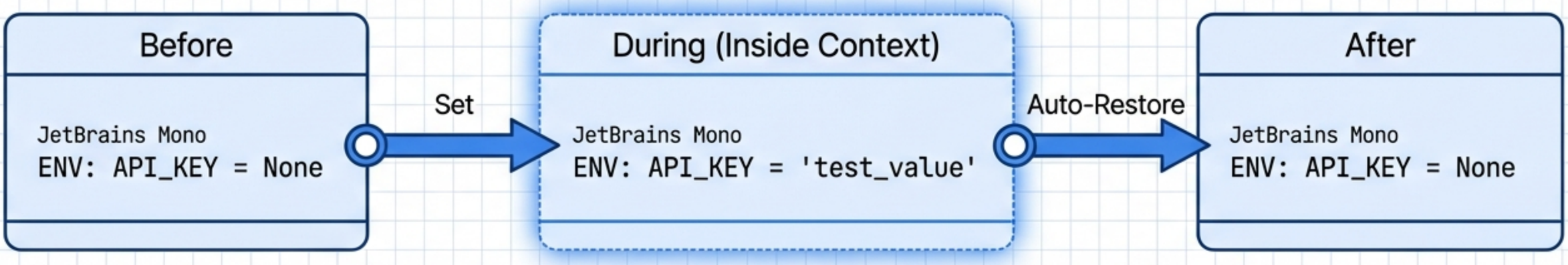
Temp_Folder

```
Temp_Folder(temp_files_to_add=0)
```

- **Bulk Creation:** Generate N files instantly with `temp_files_to_add`.
- **Navigation:** `files_in_folder()` lists contents.
- **Absolute Paths:** `files(show_parent_folder=True)`.



Controlling the Invisible: Environment & Path



Temp_Env_Vars: Isolated Environment Variables

Temp_Env_Vars

- **Usage:** Pass a dictionary `{'API_KEY': 'test_value'}`.
- **Safety:** Original values restored on exit. Non-existent keys are deleted.

Temp_Sys_Path

- **Usage:** `Temp_Sys_Path('/path/to/modules')`
- **Impact:** Temporarily modifies Python's import path.
- **Note:** Be aware of `sys.modules` caching.

Design Principles: Test-First Mindset • Chainable & Nestable

Networking: The Temp Web Server

```
JetBrains Mono
with Temp_Web_Server() as server:
    print(server.url())
    # Output: http://127.0.0.1:54321
```

Random Ports

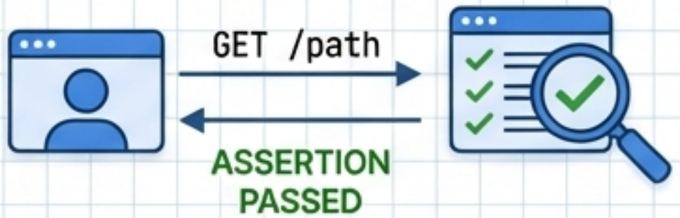
- Avoids 'Address in use' errors.
- Crucial for parallel test execution.

Safety Mechanisms

- `wait_for_stop=True`
- Ensures OS releases port before next test.

Verification

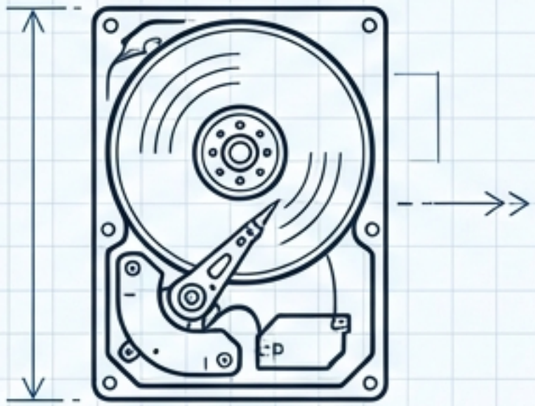
- `GET(path)` helper method.
- `GET_contains(content, path)` for assertions.



Archives: Physical & Virtual Zips

Swiss Engineering Blueprint

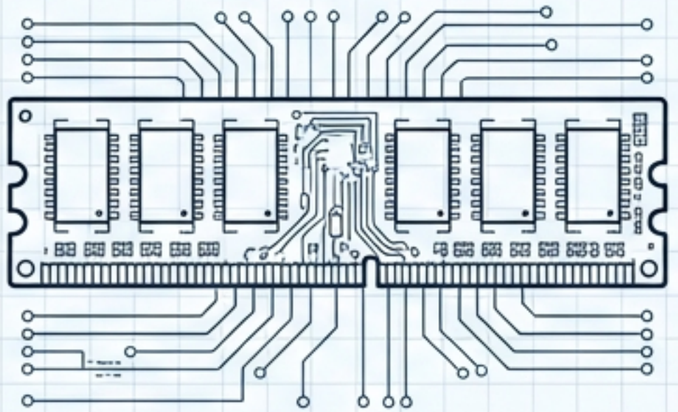
Physical (Disk)



`Temp_Zip`

- Zips a target folder to a temporary file on disk.
- **Method:** Use `move_to(target)` to simulate file transfers.

Virtual (Memory)



`Temp_Zip_In_Memory`

- Manipulate zip bytes without touching the disk.
- **Power Feature:** `add_file_from_content(path, content)` creates virtual files instantly.
- **Output:** `zip_bytes()` returns raw stream for testing.

Best Practice Note: Use `set_root_folder()` to ensure relative paths inside the archive are correct.

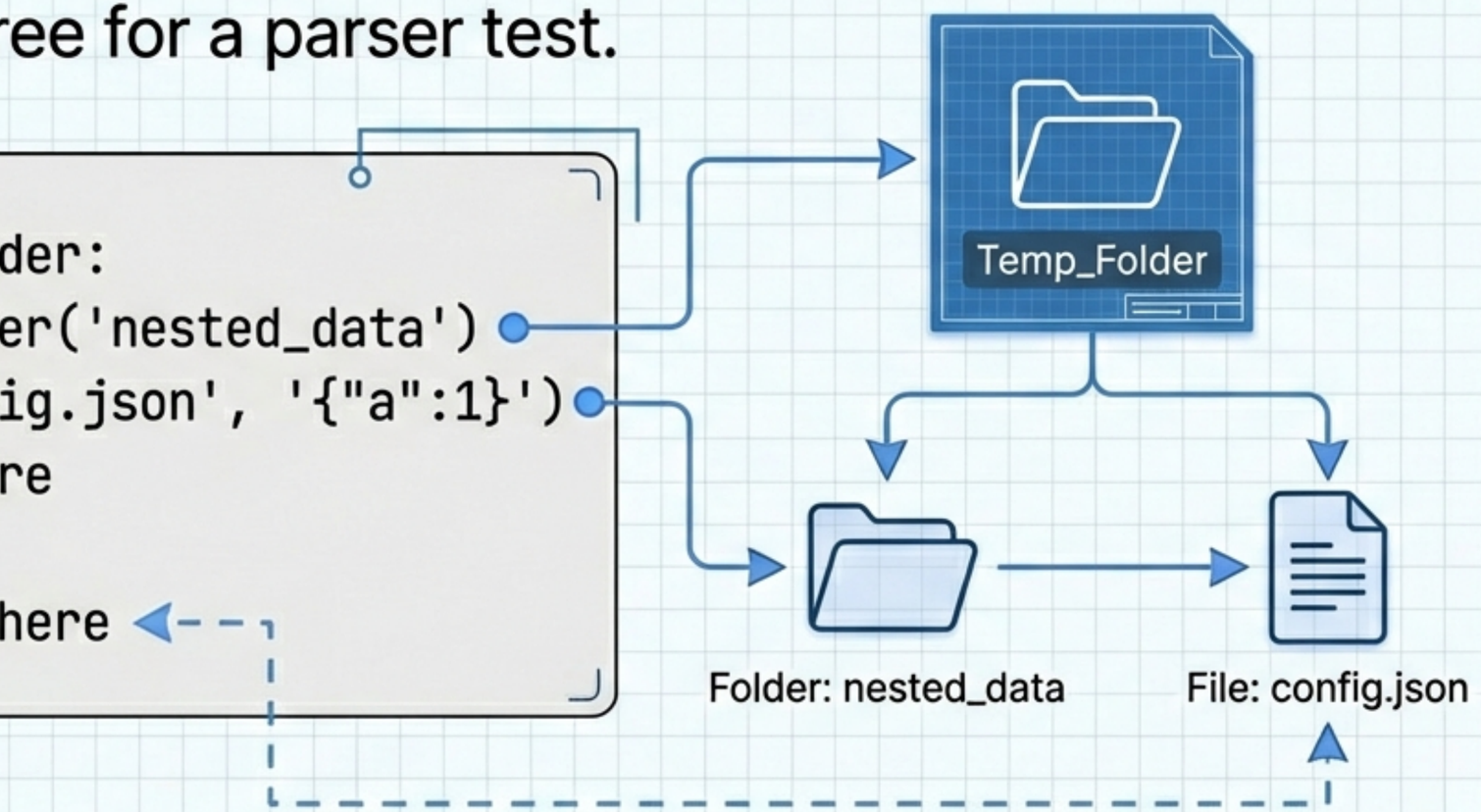
Usage Pattern I: Structure & Isolation

Creating a complex file tree for a parser test.

JetBrains Mono

```
with Temp_Folder() as folder:
    sub = folder.add_folder('nested_data')
    folder.add_file('config.json', '{"a":1}')
    # Structure exists here

    # Structure vanishes here
```

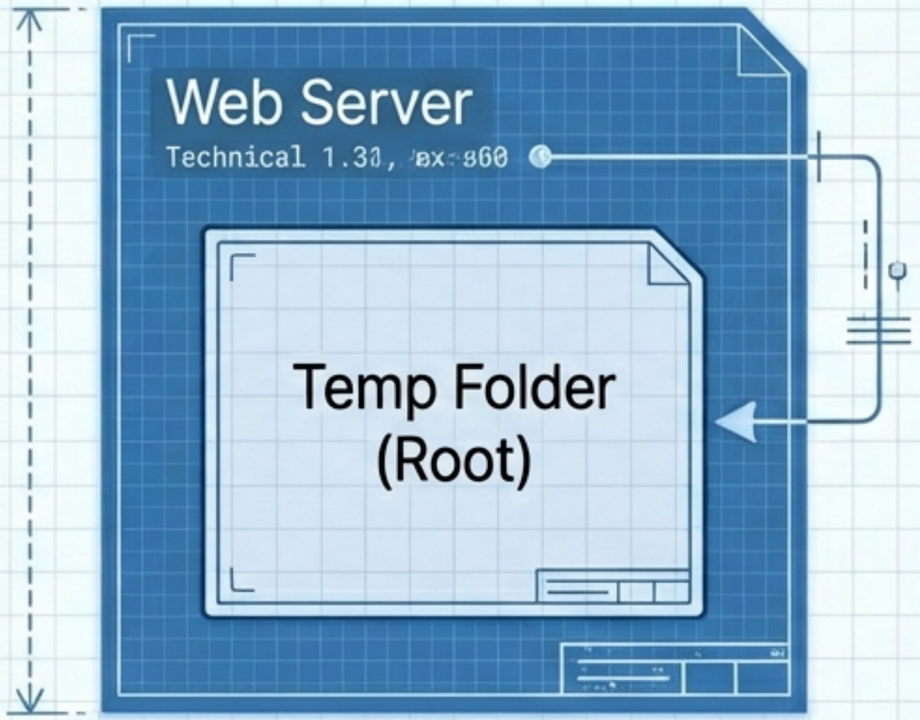


Pattern 9: Nested Folders & Bulk Creation

Usage Pattern II: Advanced Integration

Nesting Contexts

- Combine tools for full stack simulation.
- Create a `Temp_Folder` for static assets.
- Launch `Temp_Web_Server` with `root_folder` pointed at the temp folder.



Mocking

- Inject custom behaviour.
- Pass a custom class to `http_handler`.
- Simulate 404s, 500s, or JSON responses without external dependencies.

Patterns 4, 5 & 8

Best Practice Note: Use `Temp_Web_Server(port=0)` to automatically select an available port and avoid conflicts.

The Golden Rules of Temp Utilities

DO THIS

✅ Nest contexts for complex scenarios.

✅ Use `temp_files_to_add` for bulk data generation.

✅ Accept both `Temp_Folder` objects and string paths in your functions.

AVOID THIS

❌ **CRITICAL:** Do not store references to temp paths beyond the context block.

❌ Hardcode ports in `Temp_Web_Server` (creates CI conflicts).

❌ Assume cleanup order in parallel tests.

Troubleshooting Pitfalls

Symptom	Likely Cause	The Fix
File Not Found Error	Accessing resource after `__exit__`	Move logic inside the `with` block
Port Not Releasing	Server threads closing slowly	Enable `wait_for_stop=True`
Environment Leak	Manual set without context manager	Ensure proper `with` syntax usage

The Developer’s Pre-Flight Checklist

General

- [x] Is everything wrapped in a `with` statement? (Code in JetBrains Mono)
- [x] Am I using random names/ports for parallel safety?

Files

- [x] Did I use `contents` param for initialization? (Code in JetBrains Mono)
- [x] Did I use `extension` without the dot (e.g., 'json')? (Code in JetBrains Mono)

Network & Zips

- [x] Is `wait_for_stop` needed for this integration test? (Code in JetBrains Mono)
- [x] Am I using In-Memory operations to avoid disk I/O?

The Impact: Zero-Leakage Testing

“Focus on the logic, not the logistics.”

Start Building

```
$ pip install osbot-utils
Successfully installed osbot-utils-3.69.2
```

Package: `osbot_utils.testing.*`
Repository: github.com/owasp-sbot/OSBot-Utils

System Ready.