POCO
C++ Framework

JeffMill, Mar 2023

What is it?

- C++ Class Libraries
- Built on STL.
- Portable - Windows, (Embedded) Linux, iOS, etc.
  - My sample POCO apps are ~3MB each inclusive of runtime.
- Open Source, Boost Software License (but not Boost)
- Free, as in beer *and* speech.

But STL/Boost exists!

- STL
  - Provides lots of frameworks (e.g. containers)
  - Very flexible (iterator model, etc).
  - Relatively low-level.
    - It has a string, but how do I trim or convert case?
- Boost
  - Provides higher-level abstractions
  - Very complex in design and code.
  - (IMO) Designed around concepts, not implementations.

So POCO duplicates STL?

- POCO has some classes (e.g. stream) that are *similar* to STL classes, but in all cases:
  - These classes are vastly improved.
    - File handling really stinks in STL.
  - They're built on the STL classes.
  - They interoperate with STL objects.

Ok, so what features does POCO provide?

- Smart Pointers
- Strings
- Date / Time Conversion

Just Kidding

There's **WAY WAY WAY** more…

POCO “Foundation” Package (partial list)

- Caching
- Exception Hierarchy
- Date/Time (Stopwatch, TimeSpan, TimeZone, etc.)
- Dynamic objects (Pair, Variant, etc.)
- Event System
- Filesystem (Files, Paths)
- Hashing
- Logging
- Processes
- Regular Expressions
- Shared Libraries

POCO “Packages” (partial list)

- Crypto (Certs, Ciphers)
  - RSA, ECDSA, etc.
- Data (MongoDB (NoSQL), Redis (in-memory data store), MySQL etc.)
- Encodings (DBCS, etc.)
- JSON
- JWT (JSON Web Token)
- Net (FTP, HTTP, NTLM, OAuth, etc.)
- Prometheus (monitoring, alerting)
- Util (Apps, Config, Units, Windows Registry and Services)
- XML (DOM and SAX)
- Zip Archives

POCO Evaluated

Pros:

- Decent documentation
- Super clean source code.
- Seems to be written by client devs, and not research scientists.
- Built on top of existing libraries (zlib, PCRE, etc.)
- Great VCPKG support.

Cons:

- Not many samples
  - github and gist exist
- Template errors never fun.
  - Typically first error indicates problem.
- No async APIs.
  - Might be able to work around?

Cross-Platform Example

```cpp
#include <iostream>
#include <Poco/Path.h>

int main()
{
    Poco::Path path{"temp/info.txt"};
    std::cout << "Path: " << 
path.absolute().toString() << std::endl;
    std::cout << "Parent: " << 
path.absolute().parent().toString() << 
std::endl;
}
```

Ubuntu

```
$ pwd
/home/jeff/Repos/poco
$ build/redirection
Path: /home/jeff/Repos/poco/temp/info.txt
Parent: /home/jeff/Repos/poco/temp/
```

Windows

```
PS> (Get-Location).Path
C:\Users\jeffmill\Repos\poco
PS> .\build\debug\redirection.exe
Path: C:\Users\jeffmill\Repos\poco\temp\info.txt
Parent: C:\Users\jeffmill\Repos\poco\temp\
```

Calculate SHA256 sum of a file (full path). Print hex hash and filename.

```
Poco::Path path{ "/tmp/filename.bin" };
FileInputStream stream{ path.toString() };
SHA2Engine engine{ Poco::SHA2Engine::SHA_256 };
DigestInputStream dstream(engine, stream);
NullOutputStream ostream;
StreamCopier::copyStream(dstream, ostream);
std::cout << SHA2Engine::digestToHex(engine.digest)
    << " " << path.getFilename();
```

Just way too much stuff

So let’s just look at some code I wrote.

Questions?

Ask away.