

POCO C++ Libraries

Introduction and Overview

"Without a good library, most interesting tasks are hard to do in C++; but given a good library, almost any task can be made easy."

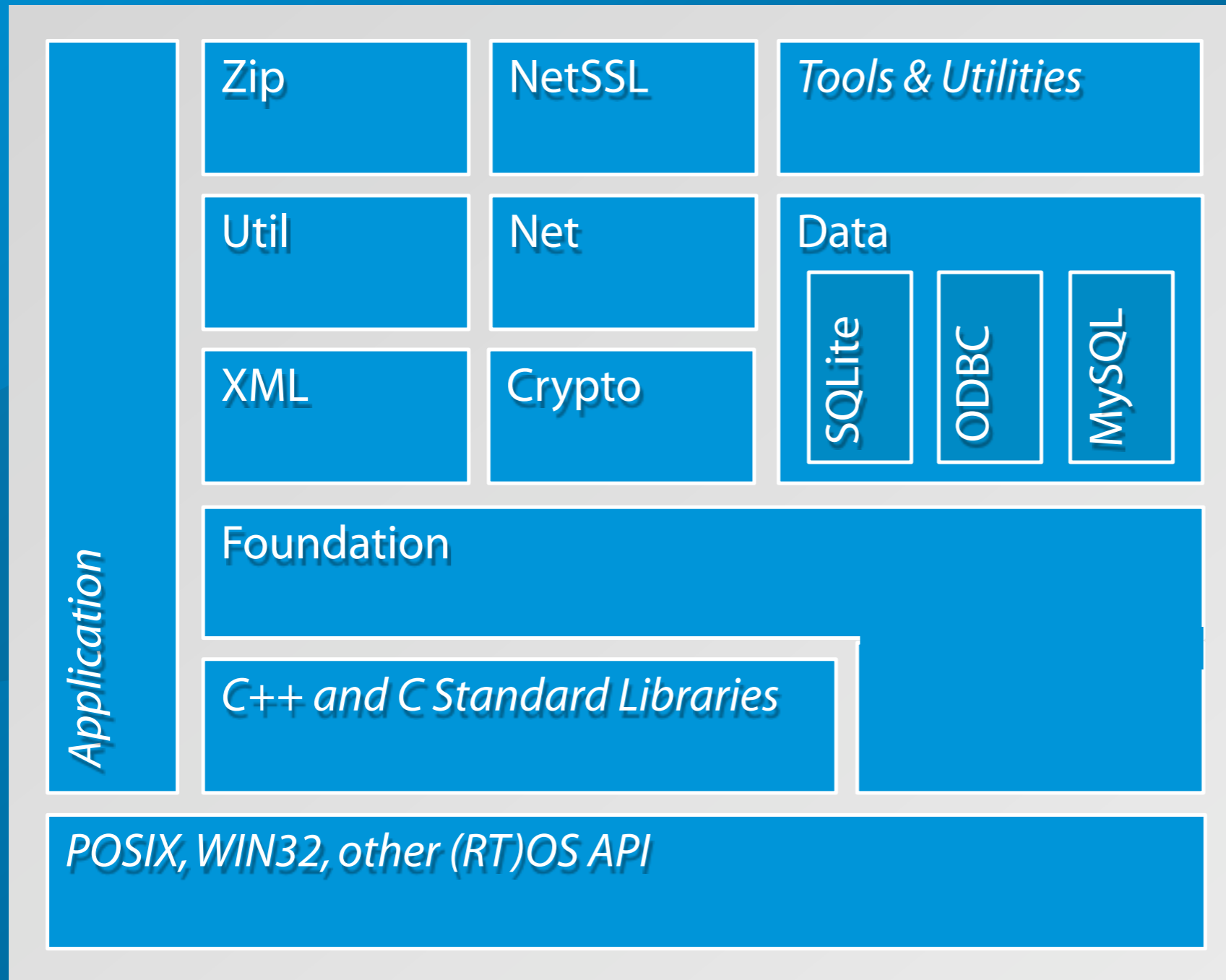
Bjarne Stroustrup

(designer and original implementor of C++)

The POCO C++ Libraries are...

- > a collection of C++ class libraries, similar in concept to the Java Class Library, the .NET Framework or Apple's Cocoa;
- > focused on "internet-age" network-centric applications;
- > written in efficient, modern ANSI/ISO Standard C++ and based on the C++ Standard Library/STL;
- > highly portable and available on many different platforms;
- > Open Source, licensed under the Boost Software License,
- > and thus completely free for both commercial and non-commercial use.

POCO C++ Libraries Overview



Features

- > Any and DynamicAny classes
- > Cache framework
- > Cryptography (cryptographic hashes, encryption based on OpenSSL)
- > Date and Time classes
- > Events (signal/slot mechanism) and notifications framework
- > FTP client for transferring files
- > Filesystem classes for platform-independent path manipulation, directory listing and globbing
- > HTML form handling

Features (cont'd)

- > HTTP server and client (also secure), C++ Server Page Compiler
- > Logging framework
- > Multithreading: basic threads and synchronization and advanced facilities (thread pool, active objects, work queues, etc.)
- > POP3 client for receiving mail
- > Platform Abstraction: write once, compile and run on multiple platforms
- > Processes and IPC
- > Reactor framework
- > Regular expressions (based on PCRE)

Features (cont'd)

- > SMTP client for sending mail
- > SQL database access (SQLite, MySQL, ODBC)
- > SSL/TLS support based on OpenSSL
- > Shared library and class loading
- > Smart pointers and memory management (buffer, pool)
- > Sockets and raw sockets
- > Stream classes for Base64 and HexBinary encoding/decoding, compression (zlib), line ending conversion, reading/writing to memory, etc.

Features (cont'd)

- > String formatting and string utilities
- > TCP server framework (multithreaded)
- > Text encodings and conversions
- > Tuples
- > URI handling
- > UTF-8 and Unicode support
- > UUID handling and generation
- > XML parsing (SAX2 and DOM) and XML generation
- > Zip file manipulation

POCO Objectives and Mission

- > POCO is a powerful, yet easy to use platform to build your applications upon
- > POCO allows you to build highly portable applications (write once – compile and run anywhere)
- > POCO is modular and scalable from embedded to enterprise applications (you only pay for what you use)
- > POCO provides consistent, comprehensive and comprehensible programming interfaces
- > POCO is written in fast, efficient C++

Objectives and Mission (cont'd)

- > POCO favors simplicity over complexity ("as simple as possible, but not simpler")
- > POCO aims for consistency in design, coding style and documentation
- > POCO emphasizes source code quality, in terms of readability, comprehensiveness, consistency, style and testability
- > POCO aims to make C++ programming fun again

Guiding Principles

- > Strong focus on **code quality, style, consistency** and **code readability** –all code must satisfy our coding styleguide (and it works – we frequently get compliments on our code quality)
- > Strong focus on tests (automated unit tests with high coverage)
- > Favor pragmatic and elegant design over "solving all the worlds problems" (if we can satisfy 95 % of all use cases with an elegant solution, and the remaining 5 % would require an overly complex design, we focus on the 95 %)
- > Build on top of solid foundations – use existing proven C libraries (e.g., expat, zlib, PCRE, SQLite) where it makes sense

History

- > Summer 2004: Günter Obiltschnig started development
- > February 2005: First release on SourceForge (Release 0.91 under Sleepycat license)
- > May 2005: First contributions by Aleksandar Fabijanic
- > January 2006: Release 1.0
- > March 2006: Release 1.1
- > July 2006: Moved to Boost license, POCO Community Website
- > August 2006: Release 1.2
- > May 2007: Release 1.3
- > July 2010: Stable Release 1.3.7, about 20 contributors, used in 100s of projects

Supported Platforms

- > Microsoft Windows
- > Linux
- > Mac OS X
- > HP-UX, Solaris, AIX*
- > Embedded Linux (uClibc, glibc)
- > iOS
- > Windows Embedded CE
- > QNX

* requires patches

POCO Usage Examples

- > building automation middleware and devices
- > industrial automation and industrial equipment
- > traffic control systems
- > healthcare applications
- > measurement, data acquisition and test systems
- > consumer electronics/home automation
- > smart metering
- > air traffic management systems
- > VoIP
- > ticketing and entrance-control systems
- > shrink-wrapped applications

Some Companies using POCO

- > **454 Life Sciences (Roche)**
using POCO in a new high-speed genome sequencer
- > **ACTIA Automotive**
using POCO and OSP in automotive diagnostic systems
- > **Appcelerator Titanium**
using POCO in its platform for Web-based desktop applications
- > **CACE Technologies**
using POCO in its Pilot product for network monitoring/analysis
- > **CodeLathe Tonido**
using POCO in its Tonido web application platform and device

Some Companies using POCO (cont'd)

- > **Comact Optimisation**
using POCO in sawmill equipment running under QNX **Echo³⁶⁰**
(EchoSystem)
using POCO in a distributed learning platform
- > **HORIBA**
using POCO in automotive test systems
- > **Novonics Corporation**
using POCO in a distributed simulation library used by US DoD
and DHS customers
- > **Nucor Steel**
using POCO in beam mill automation applications

Some Companies using POCO (cont'd)

- > **Schneider Electric Buildings Business (TAC)**
using POCO in new building automation platform (running on Embedded Linux devices and Windows/Linux servers)
- > **StreamUnlimited**
using POCO in embedded applications for TV set-top boxes
- > **Starticket**
using POCO in a ticketing/entrance control system running on an embedded Linux (OpenEmbedded on XScale) platform

POCO – Scalability Embedded

- > POCO is well-suited for embedded systems running under Embedded Linux, Windows Embedded CE or QNX.
- > POCO-based applications (using the built-in web server) run on 75 MHz ARM9-based Linux systems (uClibc) with 8 MB RAM and 4 MB Flash (e.g. Digi Connect ME 9210).
- > A typical POCO-based application using the web server from the Net library has a statically linked size of 2 MB and uses about 2 – 3 MB of RAM.
- > Typical mid-ranged embedded platforms (32 – 64 MB RAM, 16 – 64 MB Flash, 180 MHz ARM9) provide plenty of resources even for more complex applications (using OSP and Remoting).



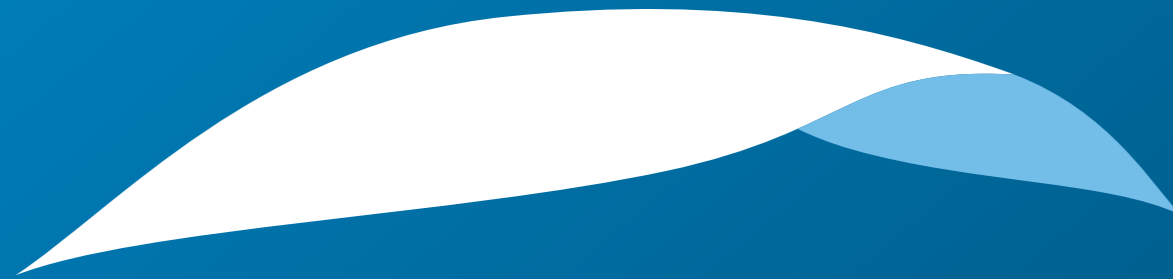
POCO and Embedded – Code Size

- > POCO (including SSL/Crypto) libraries use less than 4 MB of Flash storage (compressed jffs2 or squashfs).
- > The RAM overhead for such an application is below 8 MB.

```
guenter@cis-digiel:~/ws/poco-1.3$ ls -l lib/Linux/armv5tejl/*.so.*
-rwxr-xr-x 1 guenter guenter 103752 2009-03-03 19:12 lib/Linux/armv5tejl/libPocoCrypto.so.6
-rwxr-xr-x 1 guenter guenter 1582720 2009-03-03 18:43 lib/Linux/armv5tejl/libPocoFoundation.so.6
-rwxr-xr-x 1 guenter guenter 907192 2009-03-03 18:47 lib/Linux/armv5tejl/libPocoNet.so.6
-rwxr-xr-x 1 guenter guenter 293960 2009-03-03 19:11 lib/Linux/armv5tejl/libPocoNetSSL.so.6
-rwxr-xr-x 1 guenter guenter 281048 2009-03-03 18:45 lib/Linux/armv5tejl/libPocoUtil.so.6
-rwxr-xr-x 1 guenter guenter 577588 2009-03-03 18:44 lib/Linux/armv5tejl/libPocoXML.so.6
-rwxr-xr-x 1 guenter guenter 353312 2009-03-03 18:54 lib/Linux/armv5tejl/libPocoZip.so.6
```

POCO Benefits & Features

- > Comprehensive, complete and mature C++ frameworks that save lots of work and help bringing the product to market sooner.
- > Easy learning curve through intuitive, consistent and comprehensible programming interfaces, lots of sample code and good documentation.
- > Native C++ code performance (no VM overhead, etc.), low memory requirements.
- > Platform independence: write once – compile and run everywhere.
 - > In many cases, a large part of an application (everything that does not need access to specific hardware) can be tested and debugged on the development host.
 - > An application can be easily ported to a new platform.



appliedinformatics

Copyright © 2006-2010 by Applied Informatics Software Engineering GmbH.
Some rights reserved.

www.appinf.com | info@appinf.com
T +43 4253 32596 | F +43 4253 32096

