Italian C++ Community

www.italiancpp.org

# C++ interoperability with other languages

Alberto Bignotti

sigeo
Servizi Informativi Geografici

ELETTRIC80

RECOGNITION ROBOTICS
The Visual Guidance Company

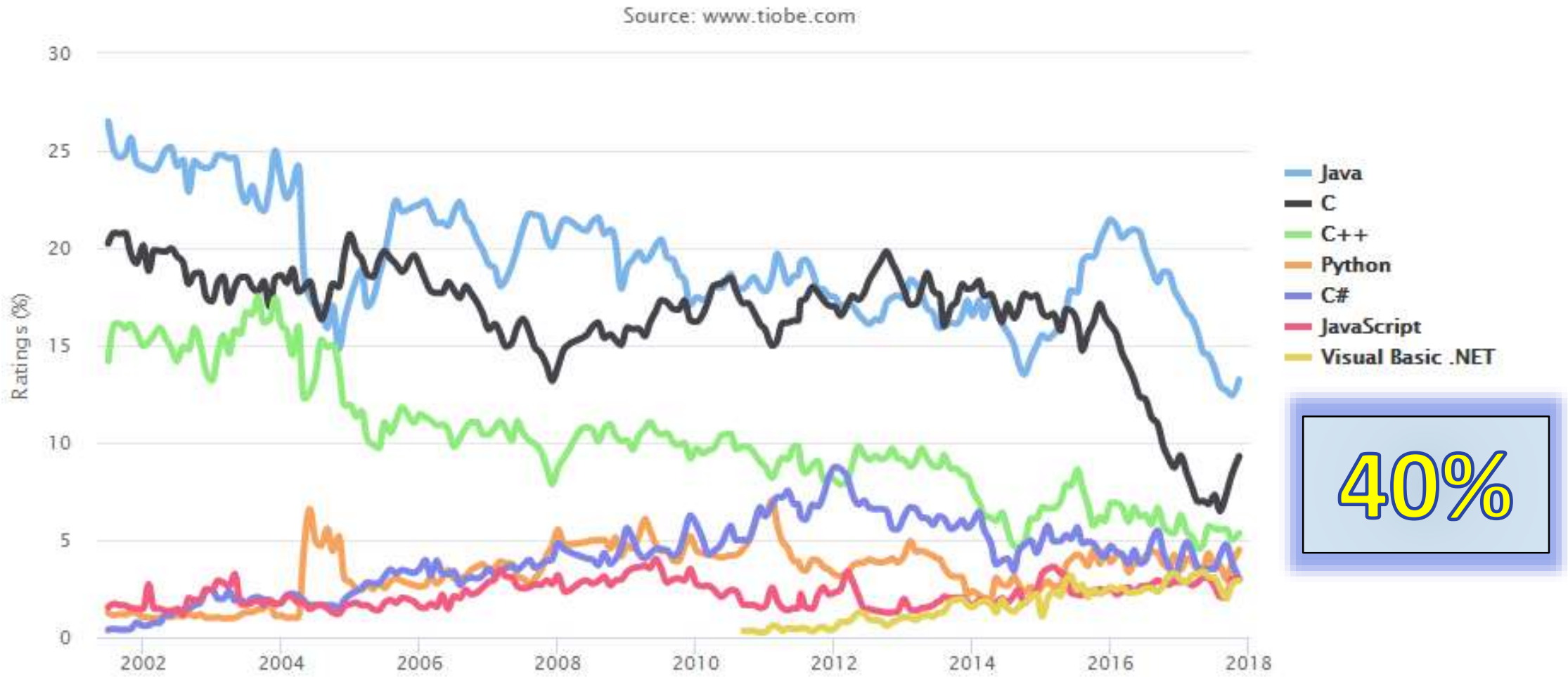
C++ Day 2017
2 Dicembre, Modena

# TIOBE Index for November 2017

The TIOBE Programming Community index is an indicator of the popularity of programming languages.

https://www.tiobe.com/tiobe-index/

# TIOBE Programming Community index



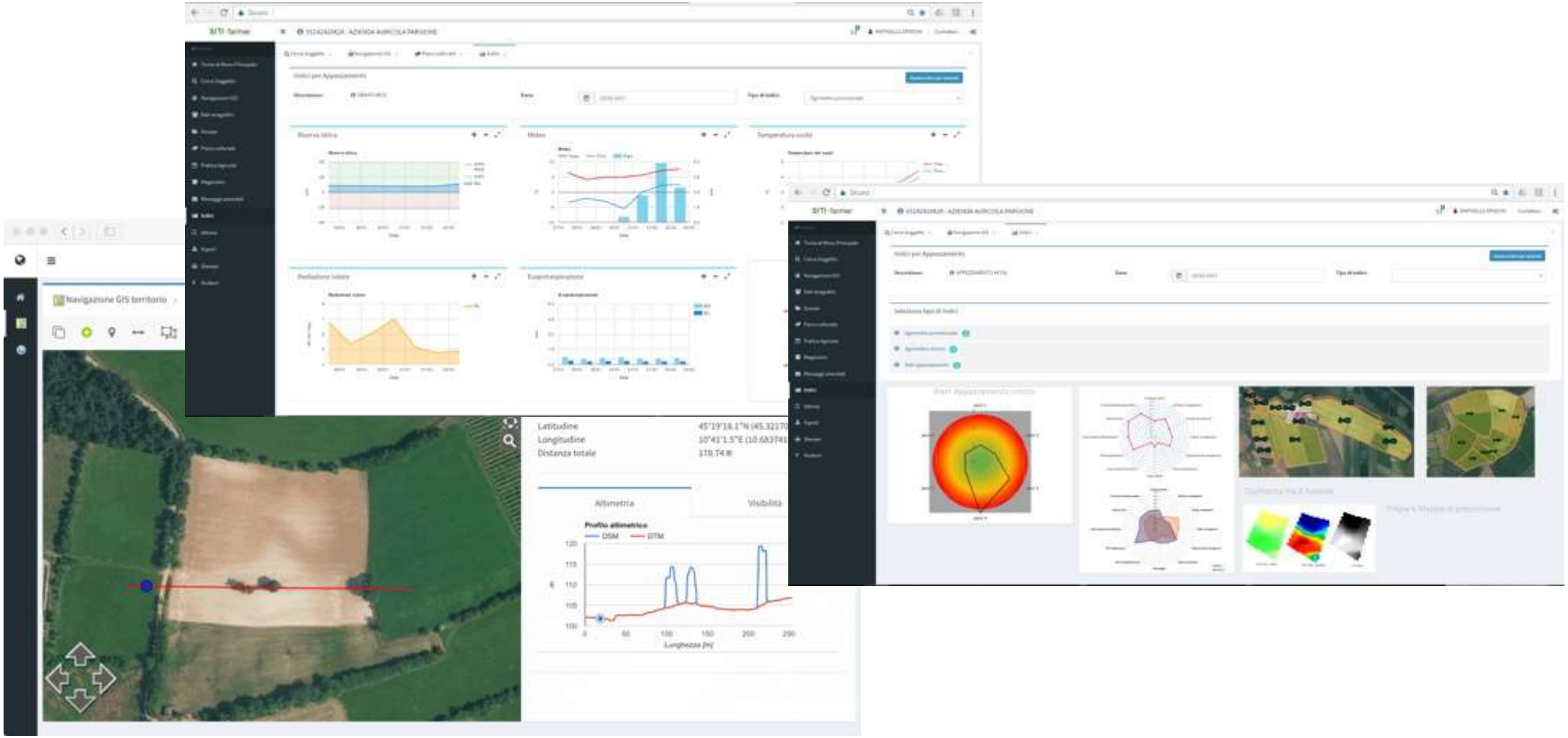
# Ispirazione

itCppCon17

Costruire un bridge C++ tra NodeJS e C# (Raffaele Rialdi)

http://italiancpp.org/itcppcon17

# Necessità


Navigazione GIS territorio
Latitudine
Longitudine
Distanza totale
Altimetria
Visibilità
Profilo altimetrico
DSM
DTM

# Il mio lavoro

Http Services Framework, Win + Linux

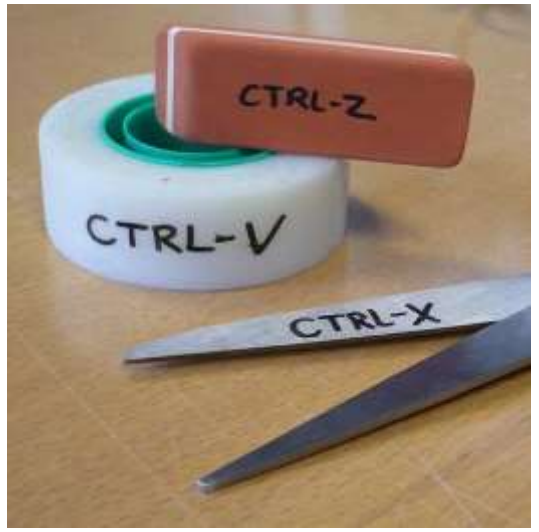
Linguaggi (C++, R, Java, SQL, PL/SQL e PL/pgSQL)

Framework di base

- Http
- Database and datasources
- Logging
- Configuration
- In memory cache
- Event notification
- Db interface
- Security
- Scalability

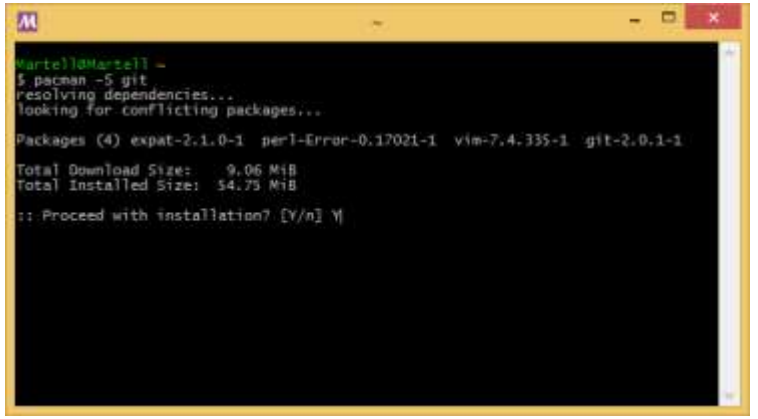
# Preparazione della presentazione

- Copia / Taglia / Incolla
- Semplificare
- Cross platform
- Librerie facili da compilare



# Preparazione della presentazione

La mia C++ toolchain (on windows)




http://www.msys2.org/

MSYS2 is a software distro

and building platform for Windows

pacman -S mingw-w64-x86_64-toolchain

pacman -S mingw-w64-x86_64-gcc

pacman -S mingw-w64-x86_64-gdb

…

C++11


https://codelite.org/

A Free, open source, cross platform C,C++,PHP and Node.js IDE

# Librerie di base

http://fmtlib.net

alternative to printf

https://zlib.net/

Compression Library

https://github.com/mity/md4c

C Markdown parser. Fast

https://github.com/gabime/spdlog

Super fast C++ logging library.

https://github.com/civetweb/civetweb

Embedded C/C++ web server

http://www.boost.org

C++ source libraries

https://github.com/skystrife/cpptoml

parsing TOML

https://curl.haxx.se/libcurl/

the multiprotocol file transfer library

https://unqlite.org/

An Embeddable NoSQL Database Engine

# Alfred

A C++ multi-threaded http REST server

I servizi possono essere implementati in

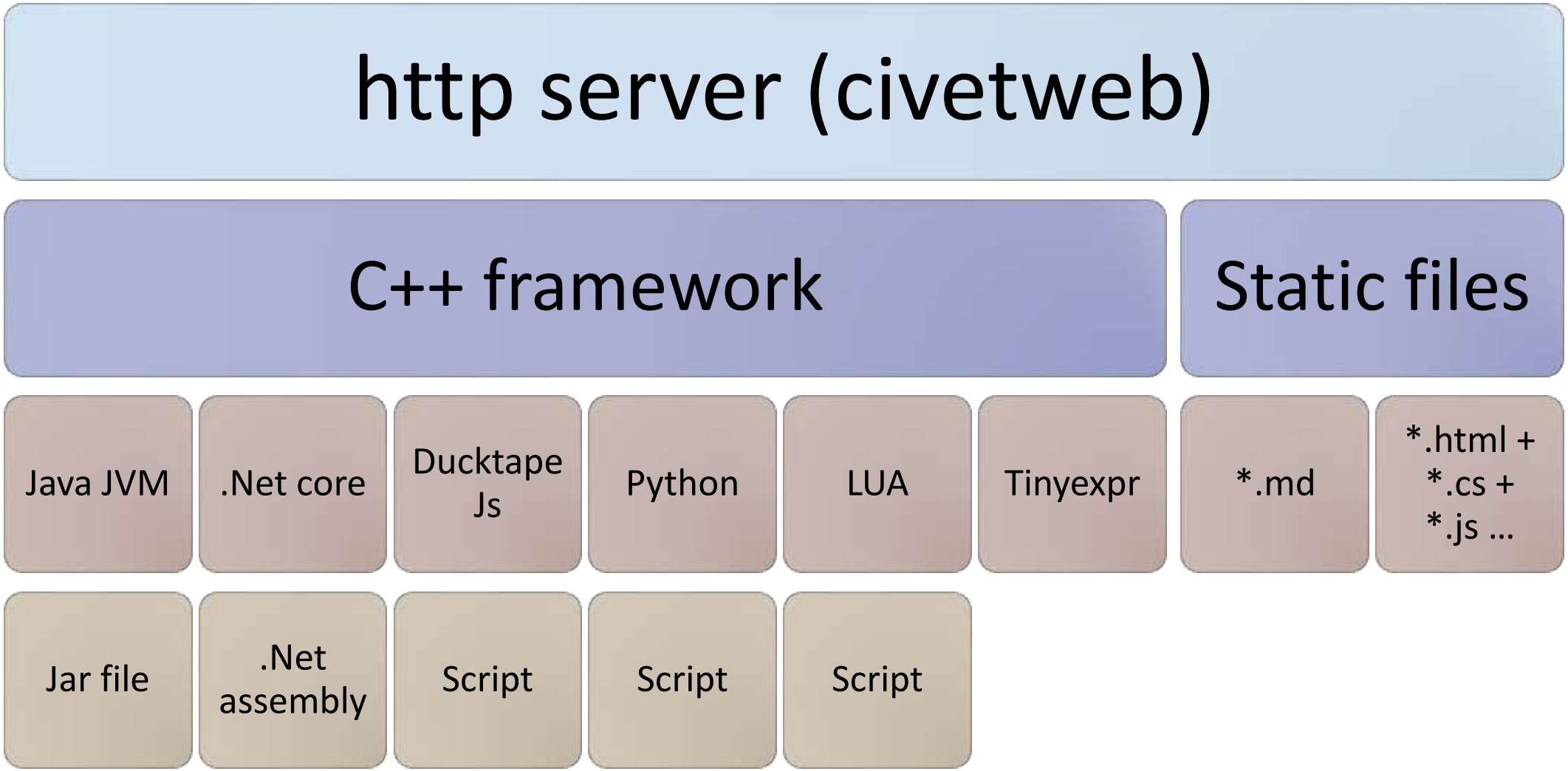
- Linguaggio matematico
- C++
- Javascript
- Lua
- Python
- C#
- Java

# Alfred

## FastCGI ?

FastCGI è un protocollo che permette di interfacciare programmi interattivi CGI con un server web. Lo scopo principale di FastCGI è quello di ottimizzare le risorse del sistema nell'interfacciamento tra il programma CGI e il server web, permettendo al server di gestire più richieste di pagina web assieme.

# Alfred


http server (civetweb)
C++ framework
Static files
Java JVM
.Net core
Ducktape Js
Python
LUA
Tinyexpr
*.md
*.html + *.cs + *.js …
Jar file
.Net assembly
Script
Script
Script

# Alfred - Civetweb

https://github.com/civetweb/civetweb

“Project mission is to provide easy to use, powerful, C/C++ embeddable web server with optional CGI, SSL and Lua support. CivetWeb has a MIT license so you can innovate without restrictions.

CivetWeb can be used by developers as a library, to add web server functionality to an existing application. It can also be used by end users as a stand-alone web server. It is available as single executable, no installation is required.”

CivetWeb has been forked from the last MIT version of Mongoose. Since 2013

# Requisiti

Da C++ chiamo <lang>

Da <lang> chiamo C++ (callback)

Il framework c++ offre funzioni di esempio:

- http.call
- Log api

Passo i parametri del servizio http a <lang>

Ottengo in output status e body (json)

# TinyExpr

https://github.com/codeplea/tinyexpr

TinyExpr is a very small recursive descent parser and evaluation engine for math expressions.

# TinyExpr

https://it.wikipedia.org/wiki/Normalized_Difference_Vegetation_Index - NDVI

Il ***Normalized Difference Vegetation Index*** (NDVI) è un semplice indicatore grafico per valutare se la zona osservata contiene della vegetazione viva.

$$\mathrm{NDVI} = \frac{(\mathrm{NIR} - \mathrm{VIS})}{(\mathrm{NIR} + \mathrm{VIS})}$$

# TinyExpr

# TinyExpr

Demo

# Java - setup


http://www.oracle.com/technetwork/java/javase/downloads/index.html

jdk-9.0.1_windows-x64_bin.exe

-> tools.zip

->-> bin/

->-> conf/

…

Apro come zip file, estraggo il contenuto di tools.zip, ottengo:

**.\libs\java\x64\bin\server\jvm.dll**

# Java - setup


https://neugens.wordpress.com/2015/02/26/debugging-the-jdk-with-gdb/

“

Hotspot uses segfaults for a number of interesting things, like deoptimise, NullPointerException etc.. Apparently, this is faster than doing specific checks and jumping around the code. This is a problem for gdb though, since it will stop every now and then to some random routines you don’t really (usually!) care about:

Program received signal SIGSEGV, Segmentation fault.

Irritating, since those are all legitimate segfaults.

To avoid that just do the following in the gdb console (or from the IDE in whatever way this is handled there):

(gdb) handle SIGSEGV nostop noprint pass

Now all the interesting work can be done without interruptions

“

# Java – Compile class


**TestService.java**

```
public class TestService {
    public void doService(HashMap serviceParameters){
        // Prints "Hello, World" in the terminal window.
        System.out.println("Hello, World");
    }
}
```

> javac TestService.java

Ottengo: **TestService.class**

# Java – Create JAR


> jar cf svc01.jar alfredTest*.class

Ottengo: **svc01.jar** (Java Archive)

# Java – JNI


La **Java Native Interface** o **JNI** è
un framework del linguaggio Java che consente al codice Java di richiamare (o essere richiamato da) codice cosiddetto "nativo ".

```
#include <jni.h>
```

# Java – Jvm startup


```
JavaVM *jvm;    // Pointer to the JVM (Java Virtual Machine)
JNIEnv *env;    // Pointer to native interface

JavaVMInitArgs vm_args;                 // Initialization arguments
JavaVMOption* options = new JavaVMOption[1];     // JVM invocation options
options[0].optionString = "-Djava.class.path=.";
vm_args.version = JNI_VERSION_1_6;    // minimum Java version
vm_args.nOptions = 1;                 // number of options
vm_args.options = options;
```

prepare loading of Java VM

# Java – Jvm startup


```
jint rc = JNI_CreateJavaVM(&jvm, (void**)&env, &vm_args);

if (rc != JNI_OK) {
      //error processing...
}

cout << "JVM load succeeded: Version ";
jint ver = env->GetVersion();
cout << ((ver>>16)&0x0f) << "."<<(ver&0x0f) << endl;

jvm->DestroyJavaVM();
```

load and initialize Java VM and JNI interface

Ready to use JVM

# Java – Call Java Method from C++


```
jclass service_class = jenv->FindClass( "package/classname" );

jmethodID service_constructor = jenv->GetMethodID(service_class, "<init>", "SIGN");
jobject service_obj = jenv->NewObject(service_class, service_constructor,
CTOR_PARAMS);

jmethodID callService = jenv->GetMethodID(service_class, "methodToCall", "SIGN");
jenv->CallObjectMethod(service_obj, callService, METHOD_PARAMS);

if(jenv->ExceptionCheck())
{
        jthrowable jt = jenv->ExceptionOccurred();
        decodeException(a_error_msg, jt);
        jenv->ExceptionClear();
}

jenv->DeleteLocalRef(service_obj);
```

C++

Construct object

Call method

Exception check

Tell GC to clear

# Java – Class File Disassembler


Come leggere le firme dei metodi

```
>javap -s TestService.class

 public void doService(java.util.HashMap);
    descriptor: (Ljava/util/HashMap;)V
```

Lo uso così (c++)

```
jmethodID callService = jenv->GetMethodID(service_class,
      "doService",
      "(Ljava/util/HashMap;)V"
);
```

# Java – Call C++ Method from Java


```
static void JNICALL alfLogInfo(JNIEnv *env, jclass clazz, jstring toLog) {
        //do something
}

static JNINativeMethod method_table[] =
{
  { (char*)("alfLogInfo"), (char*)("SIGN"), (void*) alfLogInfo },
  { (char*)("alfLogDebug"), (char*)("SIGN"), (void*) alfLogDebug },
  { (char*)("alfLogError"), (char*)("SIGN"), (void*) alfLogError }
};

jclass service_class = jenv->FindClass( “alfredTest /TestService” );

ret = jenv->RegisterNatives(service_class, method_table, method_table_size);
```

C++

# Java – Call C++ Method from Java


## Java

```
package alfredTest;

public class TestService
{
		public static native void alfLogInfo(String toLog);
		public static native void alfLogDebug(String toLog);
		public static native void alfLogError(String toLog);

		public void doService(HashMap serviceParameters){
			alfLogInfo("Jave here!");
		}
}
```

# Java


Ottima documentazione!

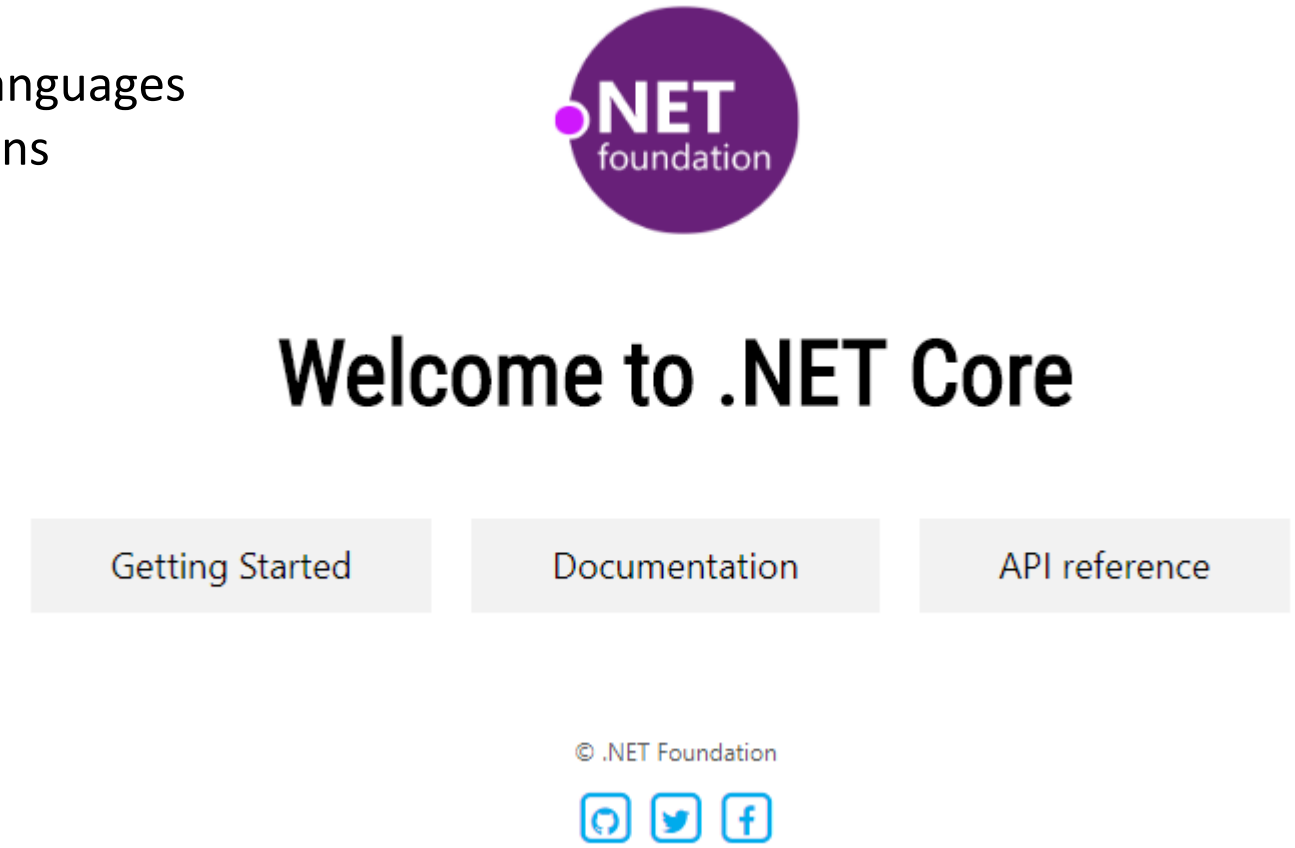
# Java


Demo

# .NET Core – C#

https://dotnet.github.io/

"The C#, Visual Basic, and F# languages can be used to write applications and libraries for .NET Core."



# .NET Core – C#


https://dotnet.github.io/

.NET Core is a general purpose development platform maintained by Microsoft and the .NET community on GitHub. It is cross-platform, supporting Windows, macOS and Linux, and can be used in device, cloud, and embedded/IoT scenarios.

https://www.microsoft.com/net/learn/get-started/windows

# .NET Core – C#


```
cd C:\dev_gcc\alfred
dotnet new classlib -o manlib
cd manlib
```

Edit *.cs

```
dotnet build
```

Per aggiungere funzioni native ho dovuto mettere nel file .csproj il seguente blocco:

```
  <PropertyGroup>
    ...
   <AllowUnsafeBlocks>true</AllowUnsafeBlocks>
    ...
  </PropertyGroup>
```

# .NET Core – C#


https://docs.microsoft.com/en-us/dotnet/core/tutorials/netcore-hosting

- Step 1 - Identify the managed entry point
- Step 2 - Find and load CoreCLR.dll
- Step 3 - Get an ICLRRuntimeHost2 Instance
- Step 4 - Setting startup flags and starting the runtime
- Step 5 - Preparing AppDomain settings
- Step 6 - Create the AppDomain
- Step 7 - Run managed code!
- Step 8 - Clean up

# .NET Core – C#


## Da C++ invoco metodo C#

```
typedef int (STDMETHODCALLTYPE * runIt)(const char* p1, int p2);
runIt pfnDelegate;

ICLRRuntimeHost2* runtimeHostPtr = (ICLRRuntimeHost2*)runtimeHost;

HRESULT hr = runtimeHostPtr->CreateDelegate(
                        domainId,
                        theAssembly.c_str(),    // Target managed assembly
                        theClass.c_str(),       // Target managed type
                        toCall.c_str(),         // Target entry point (static method)
                        (INT_PTR*)&pfnDelegate
                );

(*pfnDelegate)("alfa", 33);
```

# .NET Core – C#


## Da C# invoco metodo C++ (callback)

```
[UnmanagedFunctionPointer(CallingConvention.ThisCall)]
unsafe delegate IntPtr ptr_to_cpp_funz( int p1, string p2 );

static void doService(IntPtr cpp_callback) {
        ptr_to_cpp_funz del_to_cpp;

        del_to_cpp = Marshal.GetDelegateForFunctionPointer(cpp_callback, typeof(ptr_to_cpp_funz) );

        del_to_cpp(1, "Hello world");
}
```

# .NET Core – C#


## Callbacks (da C# chiamo C++) previste

```
enum NativeFunctions {
        curlNew,
        curlGetStatus,
        curlGetContentType,
        curlGetBody,
        curlFree,
        getParam,
        replyService,
        logInfo,
        logDebug,
        logError
};
```

# .NET Core – C#


```
void doService(){ //coreclr startup code
        (*pfnDelegate)(getNative);
}

enum DNetNativeFunctions {e_logInfo, ...};

void* getNative(int what){
        switch(what)
        {
            case e_logInfo:
               return logInfo;
            break;
            ...
        }
        return 0;
}

void logInfo(const char* toLog){
        serverLog->info(toLog);
}
```

C++

```
[UnmanagedFunctionPointer(CallingConvention.ThisCall)]
unsafe delegate IntPtr ptr_getNative( int what );
unsafe delegate void ptr_logInfo(string toLog);

enum NativeFunctions { logInfo, ...};

static void doService(IntPtr cback_getNative) {
        ptr_getNative getNative;
        ptr_logInfo del_logInfo;

        getNative = Marshal.GetDelegateForFunctionPointer(
                cback_getNative, typeof(ptr_getNative) );
        del_logInfo = Marshal.GetDelegateForFunctionPointer(
                getNative((int)NativeFunctions.logInfo), typeof(ptr_logInfo) );

        del_logInfo("Hello world");
}
```

C#

# .NET Core – C#


Demo

python™

https://www.python.it/

https://github.com/pybind/pybind11


“Seamless operability between C++11 and Python”

Python (2.7 or 3.x, or PyPy2.7 >= 5.7)

# Setup

https://www.python.it/

Installo python-3.6.3-amd64.exe

Ottengo **C:\tools\Python36**

https://github.com/pybind/pybind11

C++ 11
Header Only

Scarico e scompatto

Ottengo **C:\dev_gcc\alfred\libs\pybind11**

# pybind11 - startup

```
bool ThreadStorage::startPythonEngine(const std::string& scriptsPath) {

        pythonGuard = static_cast<void*>( new py::scoped_interpreter() );

        std::string plugInInitCode = fmt::format(
                "import sys\nimport os\nsys.path.insert(0,'{}')",
                scriptsPath
        );

        py::exec(plugInInitCode, py::globals());
        return true;
}
```

# pybind11 - shutdown

```
void ThreadStorage::stopPythonEngine() {
        delete static_cast<py::scoped_interpreter*>( pythonGuard );
}
```

# Call python method

```
PythonService pySvc;
try
{
        py::module py_module = py::module::import( scriptFileName.c_str() );
        if(isModified)
                py_module.reload();

        py::object serviceMain = py_module.attr("serviceMain");

        serviceMain( &pySvc );
}
catch(py::error_already_set& err)
{
        std::string errorMsg = err.what();
        serverLog->error("Error on {}('{}'): '{}'", __FUNCTION__, scriptFileName, errorMsg);
}
```

# Call python method

```
PYBIND11_EMBEDDED_MODULE(AlfCore, m)
{
        m.def("logInfo", &logInfo);
        m.def("logDebug", &logDebug);
        m.def("logError", &logError);
        py::class_<PythonService>(m, "AlfService")
                .def(py::init<>())
                .def("setBody", &PythonService::setBody)
                .def("setReturnCode", &PythonService::setReturnCode)
                .def("getParam", &PythonService::getParam);
        py::class_<PythonHttp>(m, "AlfHttp")
                .def(py::init<const std::string &, const std::string &, const std::string &, const std::string &, const std::string &>())
                .def("getContentType", &PythonHttp::getContentType)
                .def("getBody", &PythonHttp::getBody)
                .def("getStatus", &PythonHttp::getStatus);
}
```

# Call C++ method

```
import AlfCore
import json
import time
def serviceMain(alfRequest):
        AlfCore.logInfo( 'Python HERE! P1={}'.format( alfRequest.getParam( 'P1' ) ) )
        httpCall = AlfCore.AlfHttp('', 'https://jsonplaceholder.typicode.com/posts?userId=1', '', '', '')
        if httpCall.getStatus() == 200 and httpCall.getContentType().startswith('application/json'):
                decoded = json.loads( httpCall.getBody() )
                data = {   'user' : decoded[0]['title'],   'price' : 542.23   }
                reply = json.dumps( data )
                returnCode = 200

        alfRequest.setBody( reply )
        alfRequest.setReturnCode(returnCode)
```

# Pybind11 Alternative


# Boost.Python


# Demo

# Lua


https://www.lua.org/

Lua is a powerful, efficient, lightweight, embedd
scripting language. It supports procedural progra
object-oriented programming, functional programming,
data-driven programming, and data description.

http://luajit.org/

a **Just-In-Time Compiler** for Lua

# Lua


## Compatibility

| | | | | | |
|---|---|---|---|---|---|
| Windows | Linux | BSD | OSX | POSIX | |
| Embedded | Android | iOS | | | |
| PS3 | PS4 | PS Vita | Xbox 360 | | |
| GCC | CLANG LLVM | MSVC | | | |
| x86 | x64 | ARM | PPC | e500 | MIPS |
| Lua 5.1 API+ABI | + JIT | + BitOp | + FFI | Drop-in DLL/.so | |

redis

NGINX

# Lua - Setup


Scaricare LuaJIT-2.0.5.zip
Scompattarlo

Building with MSVC

```
cd src
msvcbuild
```

Building with MinGW

```
mingw32-make
```

Ottengo: liblua51dll.a + lua52.dll

Include dir: LuaJIT-2.0.5\src
Libs dir: LuaJIT-2.0.5\src
Additional libraries: lua51dll

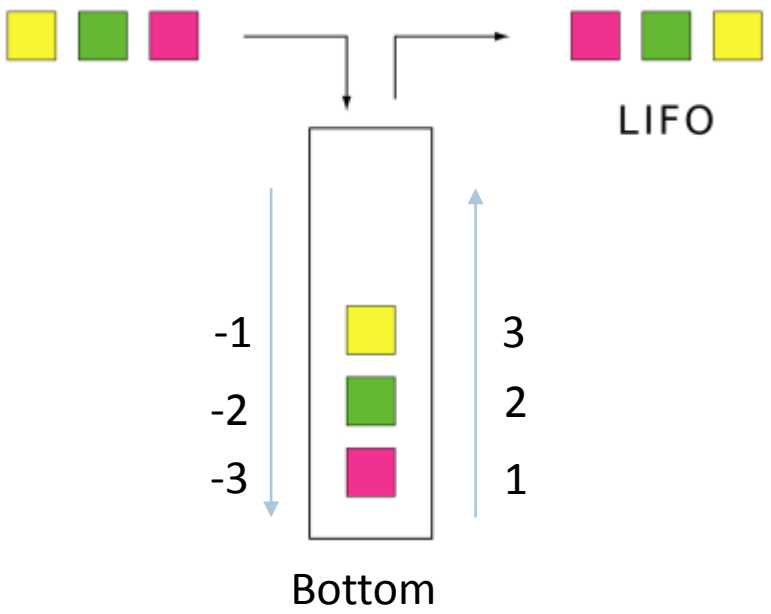
# Lua – Startup code


```
extern "C" {
        #include <lua.h>
        #include <lualib.h>
        #include <lauxlib.h>
}

int main()
{
  lua_State *state = luaL_newstate();
  luaL_openlibs( state ); // Open standard libraries
…
  lua_close(state);
  return 0;
}
```

# Lua – Call Lua function


```
int luaL_dostring (lua_State *L, const char *str);


int ret = luaL_dostring(state, scriptFile->c_str());
if(ret != 0)
{
    std::string errM = lua_tostring(state, -1);
}
lua_settop(state, 0);
```

# Lua – The lua stack




“

Whenever you want to ask for a value from Lua (such as the value of a global variable), you call Lua, which pushes the required value on the stack.

Whenever you want to pass a value to Lua, you first push the value on the stack, and then you call Lua (which will pop the value).

“

# Lua – Call C++ callback


C++

```
static int getParameters(lua_State* L)
{
   lua_newtable(L);
   for(std::map<std::string, std::string>::iterator it = curReqLua->queryParams
      it != curReqLua->queryParams.params.end();
      it++)
      {
         lua_pushstring(L, it->first.c_str());       // push key
         lua_pushstring(L, it->second.c_str());  // push value
         lua_settable(L, -3);
      }

   return 1;  // one return value
}
```

```
lua_pushcfunction(lua, getParameters);
lua_setglobal(lua, "getParameters");

lua_pushcfunction(lua, curlForLua);
lua_setglobal(lua, "curl");
```

# Lua – Call C++ callback


```
params = getParameters()
for key,value in pairs(params) do
    logInfo('Param: ' .. key .. '; Value:' .. value)
end

ret = curl("https://xyz.com", “user", “pass")
obj = json.decode(ret["body"])
```

Lua

# Lua


Demo

# Javascript

((o) Duktape

http://duktape.org/

“ Duktape is an embeddable Javascript engine, with a focus on portability and compact footprint.

Duktape is easy to integrate into a C/C++ project: add duktape.c, duktape.h, and duk_config.h to your build, and use the Duktape API to call Ecmascript functions from C code and vice versa. ”

# Javascript - Startup

((o) Duktape

```
#include <stdio.h>
#include "duktape.h"

int main(int argc, char *argv[]) {
        duk_context *ctx = duk_create_heap_default();
        duk_push_string(ctx, scriptFile->c_str());
        if (duk_peval(ctx) != 0) {
                printf("Error running: %s\n", duk_safe_to_string(ctx, -1));
        }
        duk_pop(ctx);
        duk_destroy_heap(ctx);
        return 0;
}
```

# Javascript - Call C++ callback

((o) Duktape

C++

```
static duk_ret_t native_get_parameters(duk_context *ctx) {
        alfred::ByteBuff jsonParams;
        jsonParams.append("[");
        for(std::map<std::string, std::string>::iterator it = curReq->queryParams.params.begin();
                it != curReq->queryParams.params.end(); it++)  {
                if( it != curReq->queryParams.params.begin() )  jsonParams.append(",");
                jsonParams.append( fmt::format( "{{\"{}\":\"{}\"}}", it->first, it->second );
        }
        jsonParams.append("]");
        duk_push_string( ctx, jsonParams.c_str() );
        duk_json_decode( ctx, -1 );

        return 1;  // one return value
}
```

**duk_push_c_function(duk_ctx, native_get_parameters, DUK_VARARGS);**
**duk_put_global_string(duk_ctx, "getParameters");**

# Javascript - Call C++ callback

((o) Duktape

Js

```
function executeService() {
        var params = getParameters();
        logInfo( 'P1: ' + params[0].P1 );
}
executeService();
```

# Javascript - Alternative

**Chrome V8**

**Mozilla SpiderMonkey**


**Microsoft ChakraCore**

**…**

# Javascript

((o) Duktape

Demo

# Alfred - ToDo

# Alfred

# Prossima puntata

Go

R

Linguaggio di programmazione


# Domande


@albertino80


@albertino80


italiancpp.slack.com
@albertino

albertino80@bigno.it