ИНТЕГРАЦИЯ ВИРТУАЛЬНЫХ МАШИН .NET И JAVA

DOTNEXT

Григорий Кошелев

Санкт-Петербург, 19-20 мая 2017

Идеальный слушатель

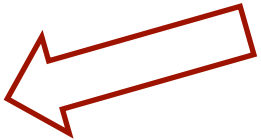
- Уже использует или хочет использовать крутые Java-библиотеки
- Хочет побольше узнать про интероп с нативным кодом
- И посмотреть, что там в Java

План (крупно)

- О задаче и решении
- Различные способы интеграции
- «Близкая» интеграция (в одном процессе)
- Детали её реализации
- Ещё больше Java интеропа
- Выводы

Дисклеймер

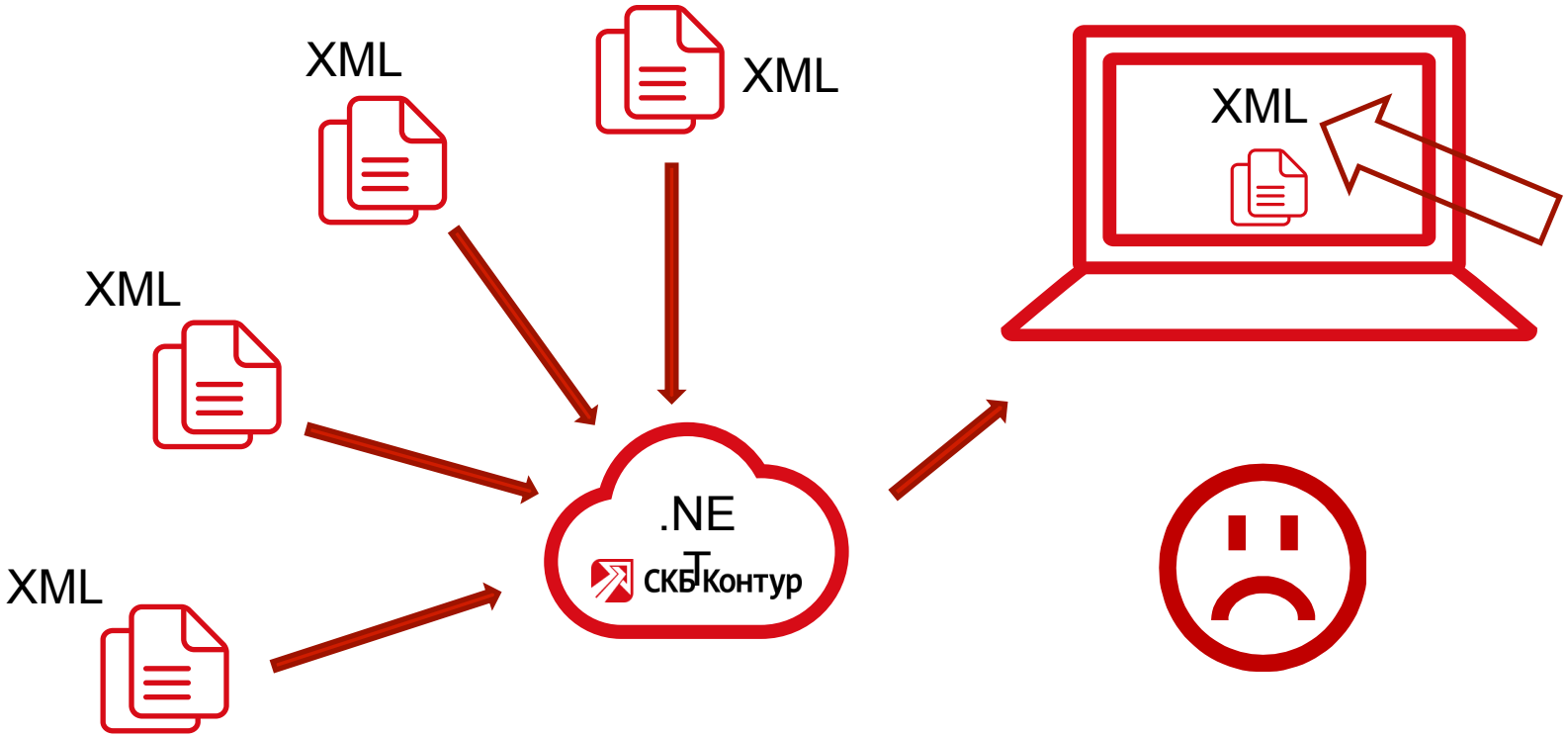
- СКБ Контур: 85% стека – .NET
- Докладчик: 85% стека – Java



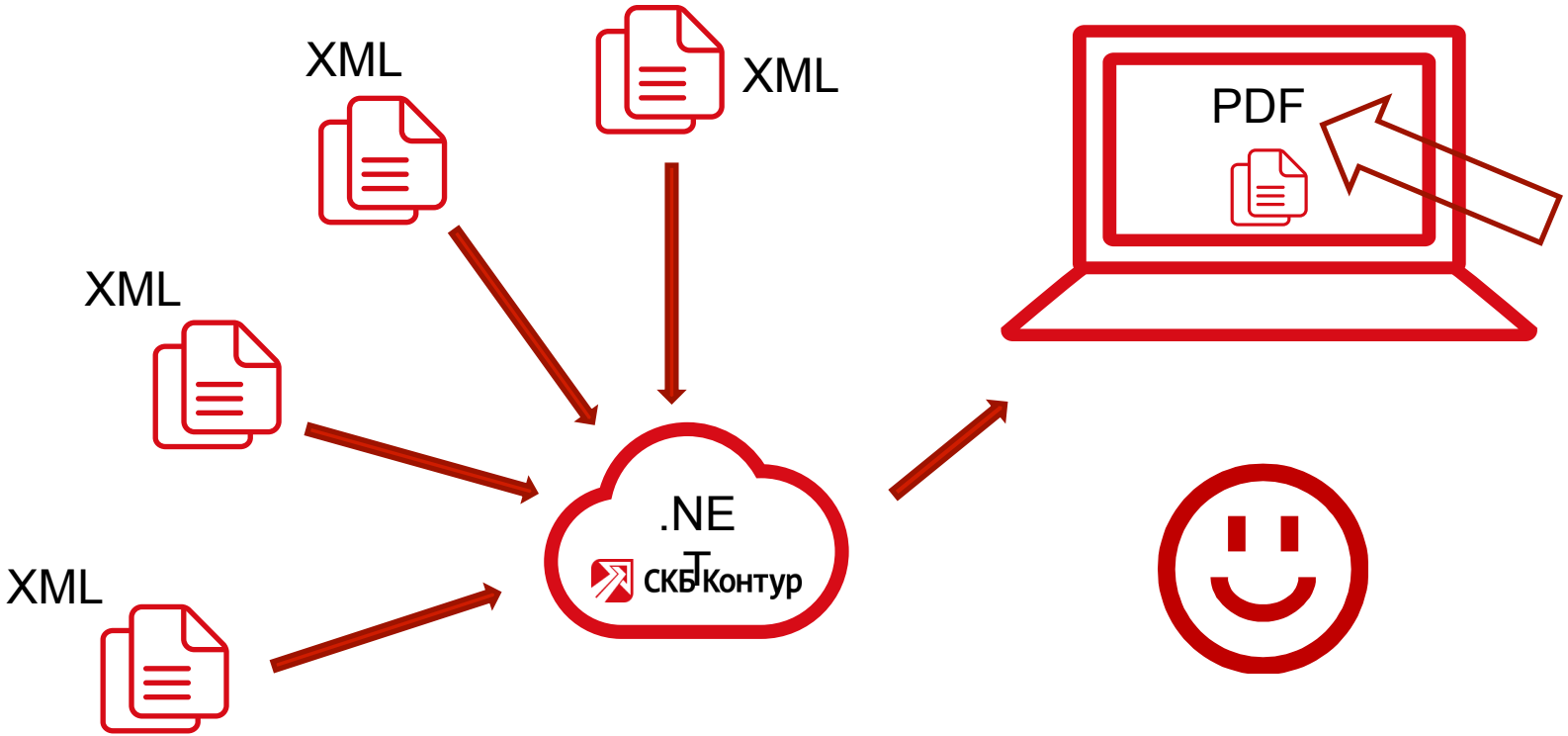
ПОЕХАЛИ?

О задаче и решении

Задача



Задача



Решение

Apache FOP (Formatting Objects Processor)

XSL-FO

PDF

Apache FOP

- Начало разработки – неизвестно
- Передана в ASF в 1999 году
- Apache FOP 1.0 – 01.07.2011
- Apache FOP 1.1 – 16.10.2012
- Apache FOP 2.0 – 02.06.2015
- Apache FOP 2.1 – 15.01.2016
- Apache FOP 2.2 – 10.04.2017

Способы интеграции

Способы (радикальные)

- **Переписать всё на C#**

Переписать всё на C#

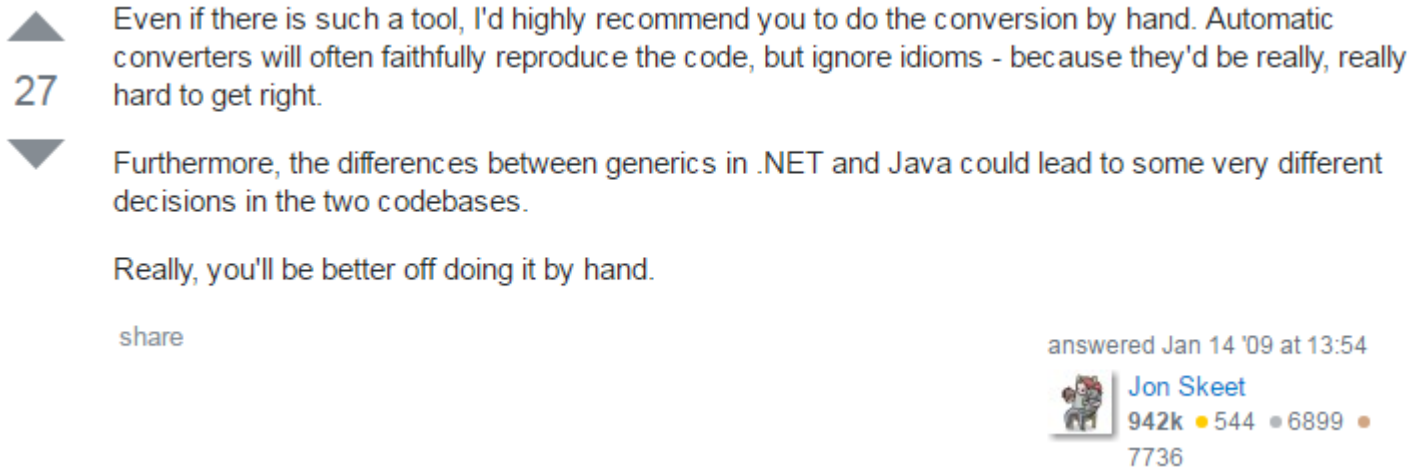
- ~ 20 лет истории
- 7900+ коммитов

Способы (радикальные)

- Переписать всё на C#
- **Сконвертировать код какой-нибудь тулзой**, а потом яростно допиливать

Конвертация кода Java -> C#

- Where can I find a Java to C# converter? (StackOverflow, 2009)



27

Even if there is such a tool, I'd highly recommend you to do the conversion by hand. Automatic converters will often faithfully reproduce the code, but ignore idioms - because they'd be really, really hard to get right.

Furthermore, the differences between generics in .NET and Java could lead to some very different decisions in the two codebases.

Really, you'll be better off doing it by hand.

share

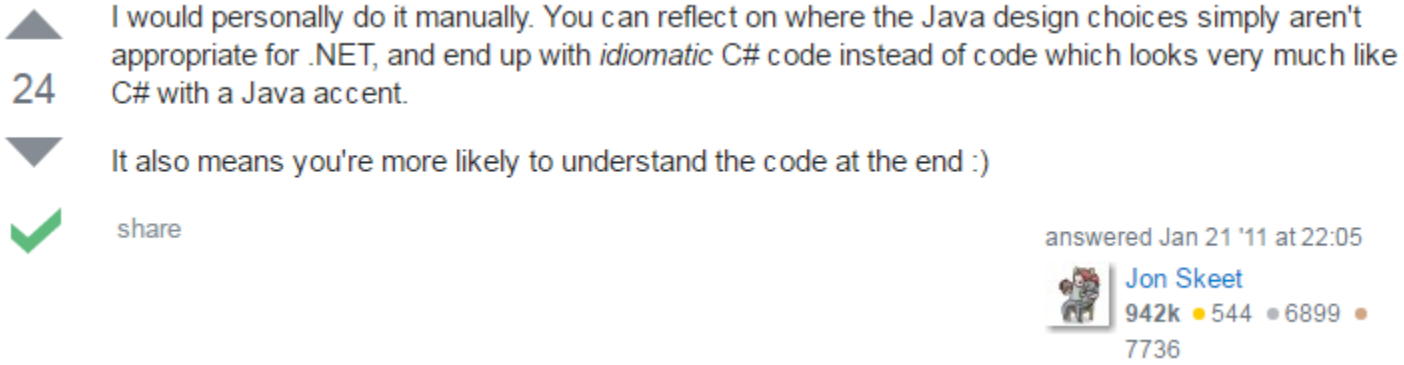
answered Jan 14 '09 at 13:54

Jon Skeet

942k • 544 • 6899 • 7736

Конвертация кода Java -> C#

- Convert Java to C# with a tool, or manually? (StackOverflow, 2011)



24

I would personally do it manually. You can reflect on where the Java design choices simply aren't appropriate for .NET, and end up with *idiomatic* C# code instead of code which looks very much like C# with a Java accent.

It also means you're more likely to understand the code at the end :)

share

answered Jan 21 '11 at 22:05

Jon Skeet

942k • 544 • 6899 • 7736

Конвертация кода Java -> C#

- Мы с Jon Skeet вас предупредили!
- Sharpen (https://github.com/mono/sharpen)
- Только Java 7
- Два года без коммитов

Способы (радикальные)

- Переписать всё на C#
- Сконвертировать код какой-нибудь тулзой, а потом яростно допиливать
- **Использовать кросс-компиляцию байткода**

Кросс-компиляция байткода

- IKVM.NET (https://www.ikvm.net/)
- Реализация JVM под .NET
- Реализация библиотеки классов Java в .NET
- Транслятор байткода (jar -> dll)
- IKVM 8.1 – 26.08.2015
- https://www.nuget.org/packages/IKVM/
- The End of IKVM.NET (21.04.2017)

Кросс-компиляция байткода

- Удачный опыт: конвертация Java-клиента для ZooKeeper и обёртки Curator; используется в продакшене
- Неудачный опыт: конвертация Apache FOP; стало работать в 2 раза медленнее, выкинули

Кросс-компиляция байткода

- Сконвертированный код «неописуемо уродлив» и требует обязательного оборачивания
- Тянет жирные библиотеки
- Не смогли завести ILMerge

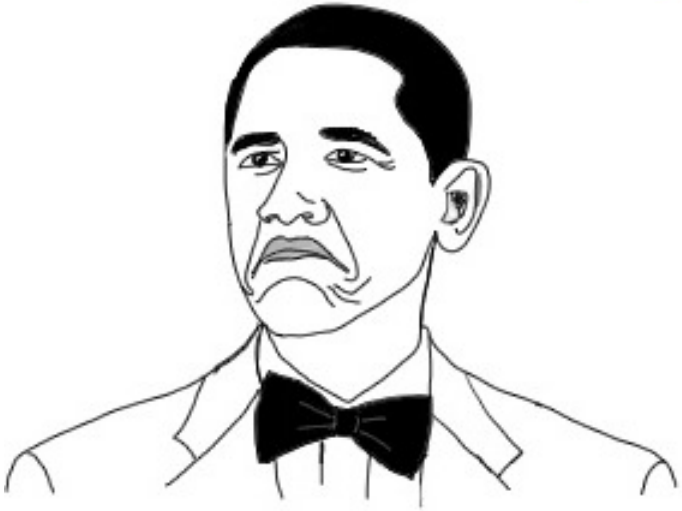
Способы (радикальные)

- Переписать всё на C#
- Сконвертировать код какой-нибудь тулзой, а потом яростно допиливать
- Использовать кросс-компиляцию байткода

- Это всё разовые операции!

Java <-> .NET

- Так сервисы же!
- Транспорт? Формат?
- HTTP REST + JSON, XML
- Свой запилить!
- Protobuf
- gRPC



gRPC

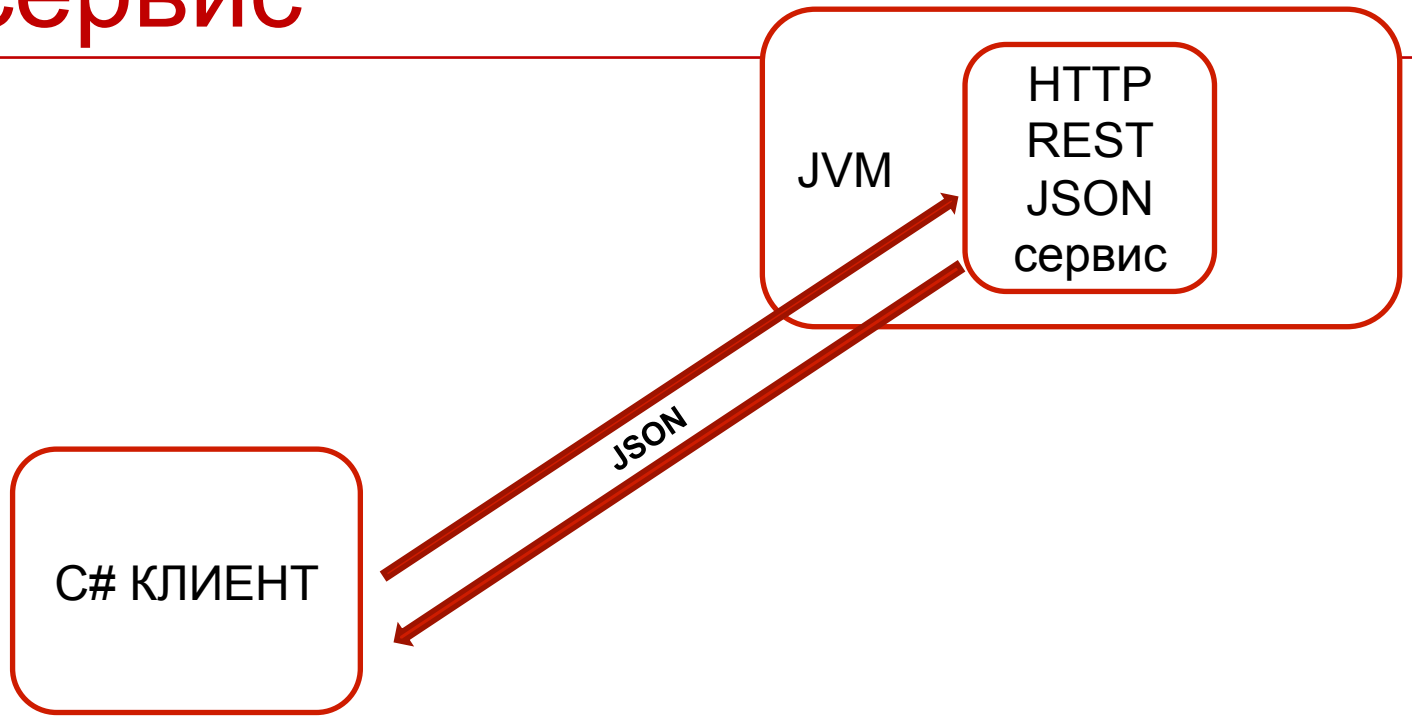
- Библиотека для удалённого вызова процедур
- Поддерживаются: C/C++, Node.js, Python, Ruby, Objective-C, PHP, C# и Java

- Первый публичный релиз – 26.02.2015
- gRPC 1.0.0 – 19.08.2016
- gRPC 1.3.2 – 11.05.2017

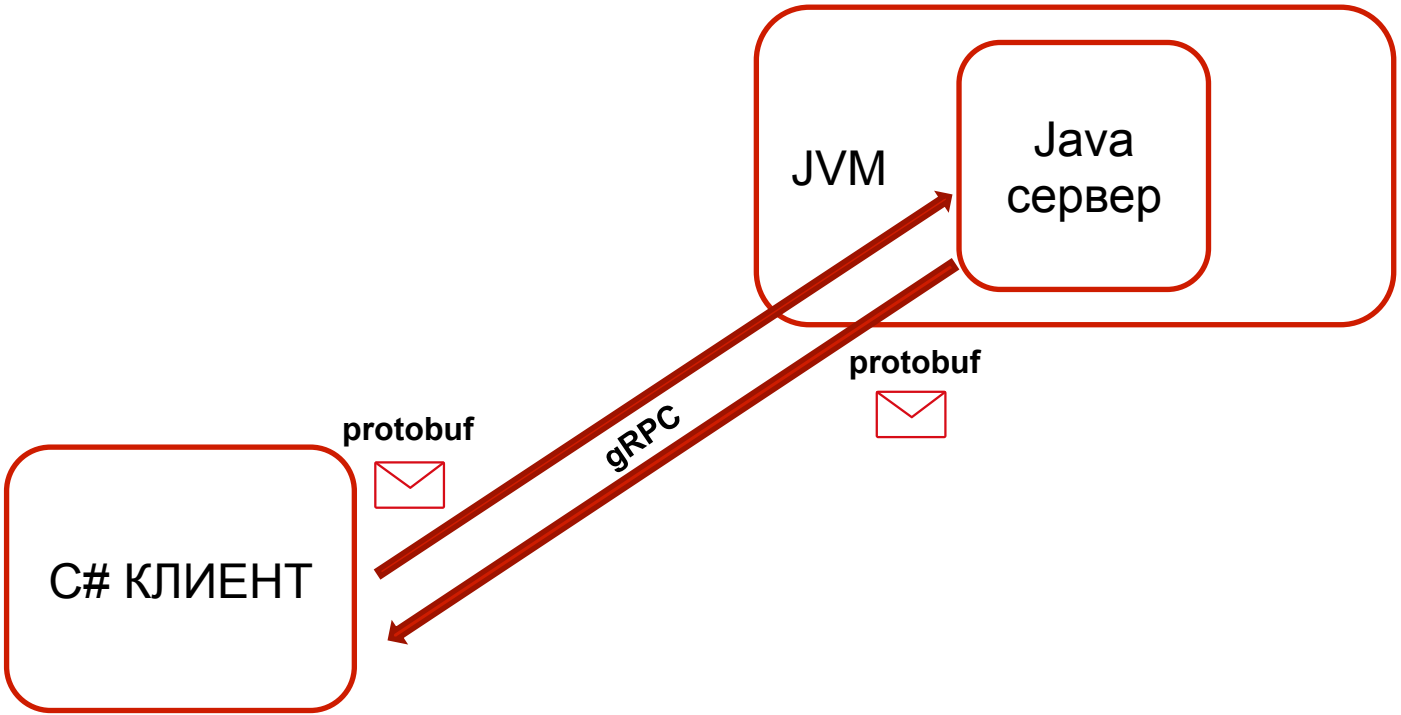
gRPC изнутри

- Protobuf 3 – описание типов данных и сериализация
- HTTP/2 в качестве транспорта
- Кодогенерация плагином для Protobuf

Java HTTP REST JSON сервис



Java gRPC сервис



Тестирование

- Intel Core i7-4710MQ, 2.5 ГГц
- 12 ГБ ОЗУ
- Windows 7 Professional x64
- Java 8 (1.8.0_112)
- .NET 4.5

Тестирование

- 10000 XML-документов (~ 1.53 ГБ)
- 1000 * 5 – на разогрев
- 9000 – на замеряемый прогон

-Xmx512m -Xms512m -Xss1m -XX:-TieredCompilation

JSON vs gRPC

	Среднее, мс	Стд. откл, мс	Относительное время
JSON	?	?	?
gRPC	?	?	?

JSON vs gRPC

	Среднее, мс	Стд. откл, мс	Относительное время
JSON	169 980	2 037	233%
gRPC	73 065	2 367	100%

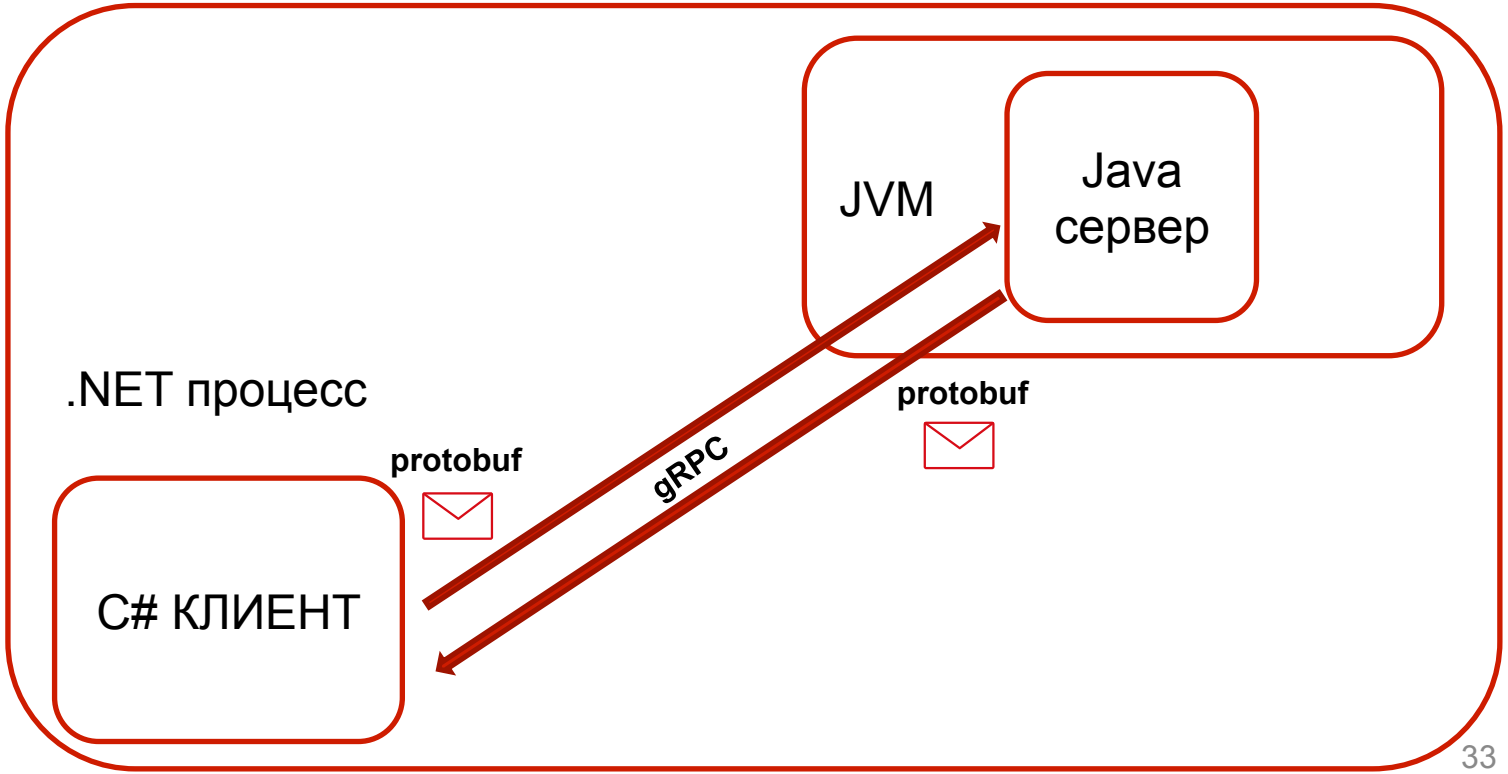
ЗАВЕРНИТЕ В ОДИН ПРОЦЕСС, ПОЖАЛУЙСТА

«Близкая» интеграция

Мотивация

- Инфраструктурные ограничения
- Перспектива снижения издержек на передачу данных

JVM внутри .NET-процесса



Java Native Interface

- Появился в 1.1
- Позволяет вызывать нативный код из Java (почти PInvoke, только страшный)
- Но это не всё! Есть Invocation API
- Позволяет вызывать Java из нативного кода

Java Native Interface

```
JavaVM *jvm;
JNIEnv *env;
JavaVMInitArgs args;
JavaVMOption* options = new JavaVMOption[1];
options[0].optionString = params;
args.nOptions = 1;
args.options = options;
args.version = JNI_VERSION_1_6;
args.ignoreUnrecognized = 0;
int result = JNI_CreateJavaVM(&jvm, (void**)&env, &args);
```

Осторожно!

C++

Java Native Interface

```
jclass programClass = env->FindClass("ru/kontur/Program");
jmethodID doSmthMethod = env->GetStaticMethodID(programClass, "doSmth",
"(ILjava/lang/String;)I");
jint intParam = …;
jstring stringParam = …;
jint result = env->CallStaticIntMethod(programClass, doSmthMethod, intParam,
stringParam);
```

Осторожно!

C++

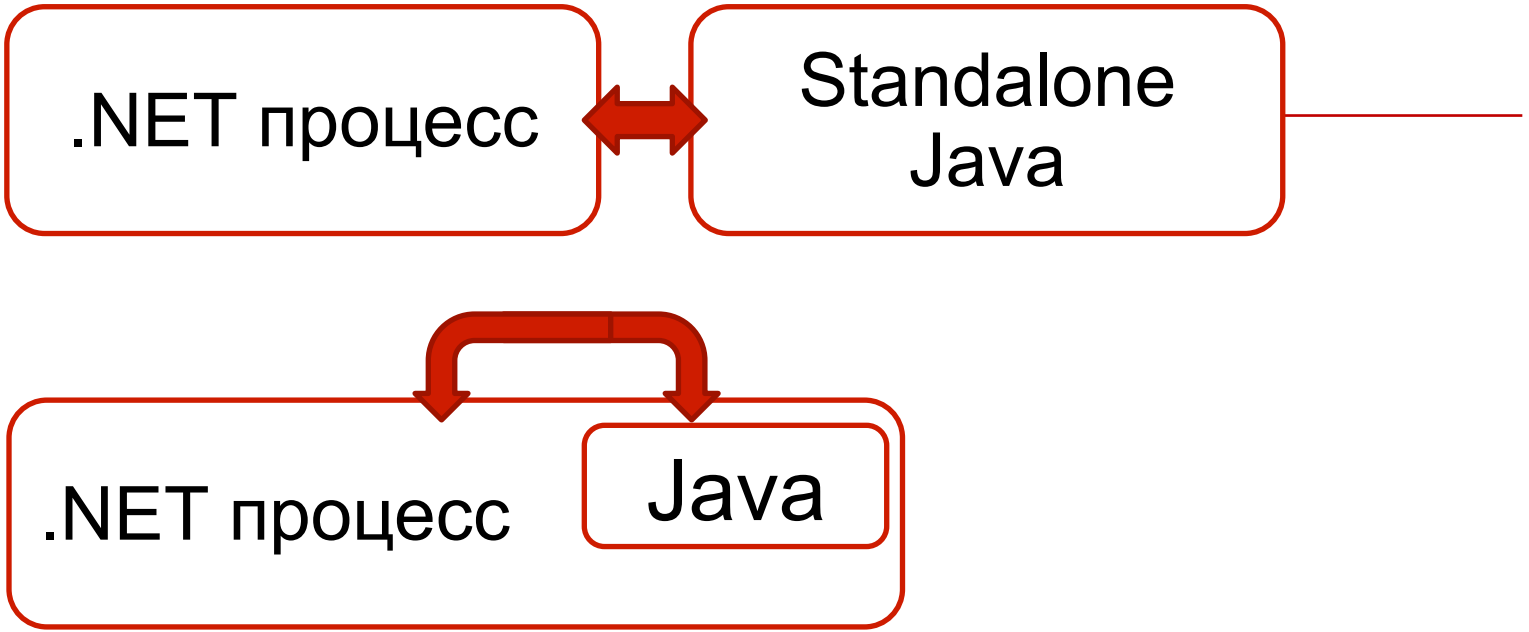
```
package ru.kontur;

public class Program {
    public static int doSmth(int x, String str) {
        return 0;
    }
}
```

C#-обёртка вокруг JNI

C#-обёртка вокруг JNI

```
JavaVmWrapper vm;
JniEnvWrapper env;
unsafe
{
    IntPtr envP; IntPtr vmP;
    CreateJavaVm(&vmP, &envP, vmArgsP);
    vm = new JavaVmWrapper(vmP);
    env = new JniEnvWrapper(envP);
}
var classPtr = env.FindClass("ru/kontur/Program");
var constructor = env.GetMethodId(classPtr, "<init>", "()V");
var runMethod = env.GetMethodId(classPtr, "run", "()V");
var obj = env.NewObject(classPtr, constructor, port);
env.CallObjectMethod(obj, runMethod);
```

gRPC снаружи vs gRPC



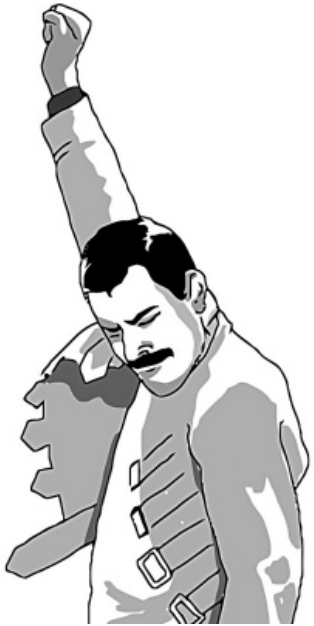
gRPC снаружи vs gRPC внутри

	Среднее, мс	Стд. откл, мс	Относительное время
Снаружи	73 065	2 367	100%
Внутри	?	?	?

gRPC снаружи vs gRPC внутри

	Среднее, мс	Стд. откл, мс	Относительное время
Снаружи	73 065	2 367	100%
Внутри	72 002	1 266	99%

gRPC снаружи vs gRPC внутри



НЕМНОГО ДЕТАЛЕЙ: КАК? ЧТО? И ГДЕ ГРАБЛИ ЛЕЖАТ?

Детали реализации

План по реализации

- Импорт Java-бинаря в .NET-процесс
- Создаём JVM внутри .NET-процесса
- Первый контакт с JVM (и первые грабли)
- Следим за ~~руками~~ ссылками
- Пробуем в многопоточность
- И другой Java-специфичный интероп

План по реализации

- **Импорт Java-бинаря в .NET-процесс**
- Создаём JVM внутри .NET-процесса
- Первый контакт с JVM (и первые грабли)
- Следим за ~~руками~~ ссылками
- Пробуем в многопоточность
- И другой Java-специфичный интероп

PInvoke и импорт JVM.dll

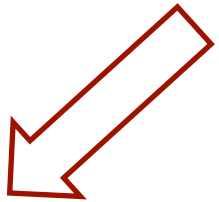
```
[DllImport("kernel32.dll", EntryPoint = "LoadLibraryW")]
private static extern IntPtr LoadLibrary(
    [MarshalAs(UnmanagedType.LPWStr)] string name);

[DllImport("kernel32.dll", EntryPoint = "FreeLibrary")]
private static extern bool FreeLibrary(IntPtr hModule);

[DllImport("kernel32.dll", EntryPoint = "GetProcessAddress")]
private static extern IntPtr GetProcessAddress(
    IntPtr hModule,
    [MarshalAs(UnmanagedType.LPStr)] string name);
```

Что там с Linux? libdl.so!

```
[DllImport("libdl.so", EntryPoint = "dlopen")]
private static extern IntPtr dlopen(
    string name, int flags);

[DllImport("libdl.so", EntryPoint = "dlclose")]
private static extern bool dlclose(IntPtr hModule);

[DllImport("libdl.so", EntryPoint = "dlsym")]
private static extern IntPtr dlsym(
    IntPtr hModule,
    string name);
```

Что там с Linux? libdl.so!

```
public static bool IsUnix
{
    get
    {
        int p = (int)Environment.OSVersion.Platform;
        return (p == 4) || (p == 6) || (p == 128);
    }
}
```



http://mono.wikia.com/wiki/Detecting_the_execution_platform

План по реализации

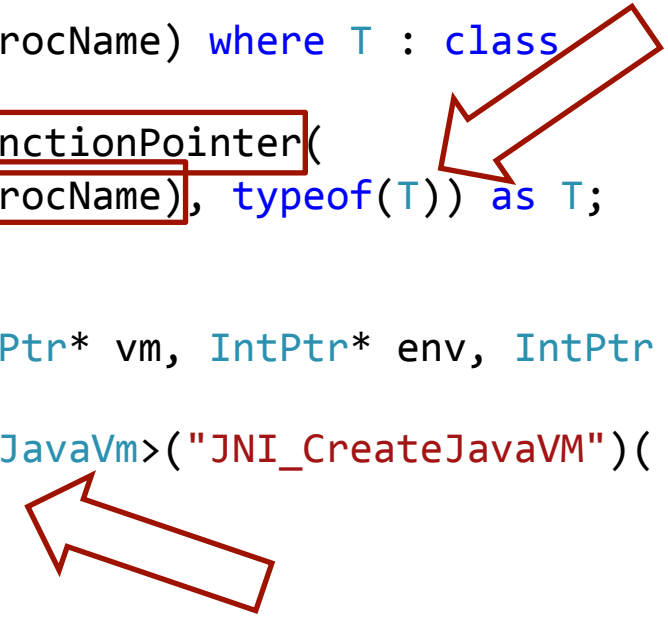
- Импорт Java-бинаря в .NET-процесс
- **Создаём JVM внутри .NET-процесса**
- Первый контакт с JVM (и первые грабли)
- Следим за ~~руками~~ ссылками
- Пробуем в многопоточность
- И другой Java-специфичный интероп

Marshal

- System.Runtime.InteropServices
- Предоставляет методы для работы с неуправляемой памятью и кодом

Процедура создания JVM

```
unsafe delegate int CreateJavaVm(IntPtr* vm, IntPtr* env, IntPtr args);

protected T GetDelegate<T>(string procName) where T : class
{
    return Marshal.GetDelegateForFunctionPointer(
        GetProcAddress(DllModule, procName), typeof(T)) as T;
}

public unsafe void CreateJavaVm(IntPtr* vm, IntPtr* env, IntPtr args)
{
    var result = GetDelegate<CreateJavaVm>("JNI_CreateJavaVM")(
            vm, env, args);
    /* process */
}
```



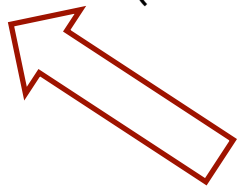
C# и struct

```
[StructLayout(LayoutKind.Sequential)]
public struct JniNativeInterface {
    /* ... */
    private readonly IntPtr GetVersion;
    private readonly IntPtr DefineClass;

    private readonly IntPtr FindClass;

    /* ... */
}
```

```
struct JNINativeInterface {
    /* ... */
    jint (JNICALL *GetVersion)(JNIEnv *env);
    jclass (JNICALL *DefineClass)
        (JNIEnv *env,
        const char *name,
        jobject loader,
        const jbyte *buf,
        jsize len);
    jclass (JNICALL *FindClass)
        (JNIEnv *env, const char *name);
    /* ... */
}
```

C# и struct

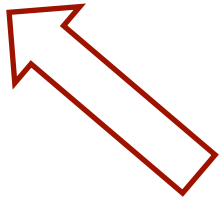
```
JniNativeInterface obj =
    (JniNativeInterface)Marshal.PtrToStructure(
        structPtr,
        typeof(JniNativeInterface));
```



C# и union

```
[StructLayout(LayoutKind.Explicit)]
public struct JValue
{
    [FieldOffset(0)] private bool booleanValue;
    [FieldOffset(0)] private byte byteValue;
    [FieldOffset(0)] private char charValue;
    [FieldOffset(0)] private short shortValue;
    [FieldOffset(0)] private int integerValue;
    [FieldOffset(0)] private long longValue;
    [FieldOffset(0)] private float floatValue;
    [FieldOffset(0)] private double doubleValue;
    [FieldOffset(0)] private IntPtr pointerVlaue;
}
```

```
typedef union jvalue {

    jboolean z;
    jbyte b;
    jchar c;
    jshort s;
    jint i;
    jlong j;
    jfloat f;
    jdouble d;
    jobject l;
} jvalue;
```



План по реализации

- Импорт Java-бинаря в .NET-процесс
- Создаём JVM внутри .NET-процесса
- **Первый контакт с JVM**
- Следим за ~~руками~~ ссылками
- Пробуем в многопоточность
- И другой Java-специфичный интероп

C#, Java и «нативные строки»

```
jclass (JNICALL *FindClass) (JNIEnv *env, const char *name);
```

```
delegate IntPtr FindClass(
    IntPtr env,
    [MarshalAs(UnmanagedType.LPStr)] string name);
```

```
delegate IntPtr FindClass(
    IntPtr env,
    [MarshalAs(UnmanagedType.LPArray)] byte[] utf);
```

```
Encoding.UTF8.GetBytes(str);
```

U+0073 Латинская буква «s» (0x73) ✔

U+041A Кириллическая буква «К» (0xD0 0x9A) ✔

U+0BF5 Символ года на тамильском «௵» (0xE0 0xAF 0xB5) ✔

U+2?218 Китайский иероглиф «?» (0xF0 0xA6 0x88 0x98)

Неуловимый баг

```
jclass (JNICALL *FindClass) (JNIEnv *env, const char *name);
```

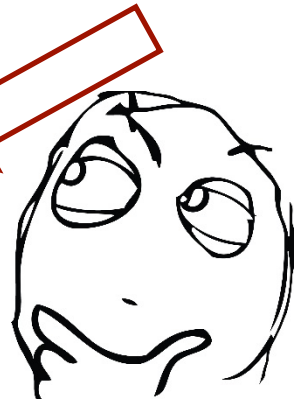
```
delegate IntPtr FindClass(
    IntPtr env,
    [MarshalAs(UnmanagedType.LPArray)] byte[] utf);
```

```
byte[] utf = Encoding.UTF8.GetBytes(str);
FindClass(env, utf);
```

Нет терминального символа «null» (0x00)!

Почему работает?

Значит, «нулевой» байт стоит следом!



План по реализации

- Импорт Java-бинаря в .NET-процесс
- Создаём JVM внутри .NET-процесса
- Первый контакт с JVM (и первые грабли)
- **Следим за ~~руками~~ ссылками**
- Пробуем в многопоточность
- И другой Java-специфичный интероп

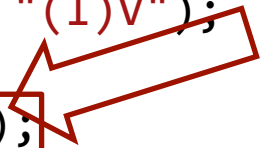
Указатели в нативную память

- Ручное управление объектами, размещёнными в нативной памяти
- Оборачиваем в IDisposable-обёртки

Объекты из Java heap

- Объект существует на стороне Java
- В нативный код передаётся по «локальной» ссылке
- Как Java GC узнает, что пора собирать?

```
var wrappedMethodId =
        Env.GetMethodId(classPtr, "<init>", "(I)V");
/* ... */
Env.DeleteLocalRef(wrappedMethodId.LocalRef);
```



План по реализации

- Импорт Java-бинаря в .NET-процесс
- Создаём JVM внутри .NET-процесса
- Первый контакт с JVM (и первые грабли)
- Следим за ~~руками~~ ссылками
- **Пробуем в многопоточность**
- И другой Java-специфичный интероп

А что с многопоточностью?

```
var env = Jvm.AttachCurrentThread();
/* process */
Jvm.DetachCurrentThread();
```

```
var globalRef = Env.NewGlobalRef(localRef);
/* process */
Env.DeleteGlobalRef(globalRef);
```

```
synchronized (obj) {
    /* synchronized block */
}
```

```
Env.MonitorEnter(obj);
    /* synchronized block */
Env.MonitorExit(obj);
```

План по реализации

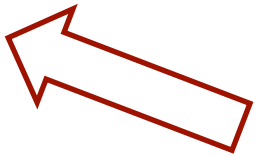
- Импорт Java-бинаря в .NET-процесс
- Создаём JVM внутри .NET-процесса
- Первый контакт с JVM (и первые грабли)
- Следим за ~~руками~~ ссылками
- Пробуем в многопоточность
- **И другой Java-специфичный интероп**

Загадка

- *Их* нет в .NET
- *Они* есть в Java (и многих бесят)
- Подсказка: бесят, потому что не дают о себе забыть

Exception

```
T SafeJniCall<T>(Func<T> func)
{
    T result = func();
    if (Env.ExceptionCheck())
    {
        var exception = Env.ExceptionOccurred();
        /* process exception */
        Env.ExceptionClear();
    }
    return result;
}
```

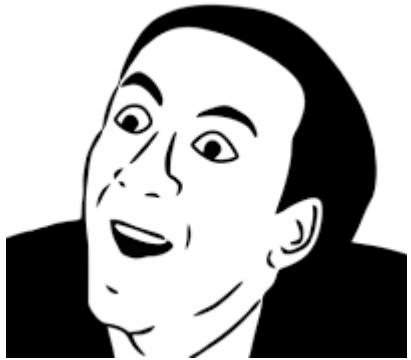


Загадка

- В .NET *они* были сразу простыми, как целочисленные типы
- В Java *они* появились только в 1.5, но ни разу не простыми
- Подсказка: я буду перечисление

Enum

```
package ru.kontur.fop;
public enum TransformationType {
    CSV, PDF, SVG;
}
```



```
var enumClass = env.FindClass("ru/kontur/fop/TransformationType");
var fieldIdPDF = env.GetStaticFieldID(enumClass, "PDF",
                 "Lru/kontur/fop/TransformationType;");
var PDF = env.GetStaticObjectField(enumClass, fieldIdPDF);
```

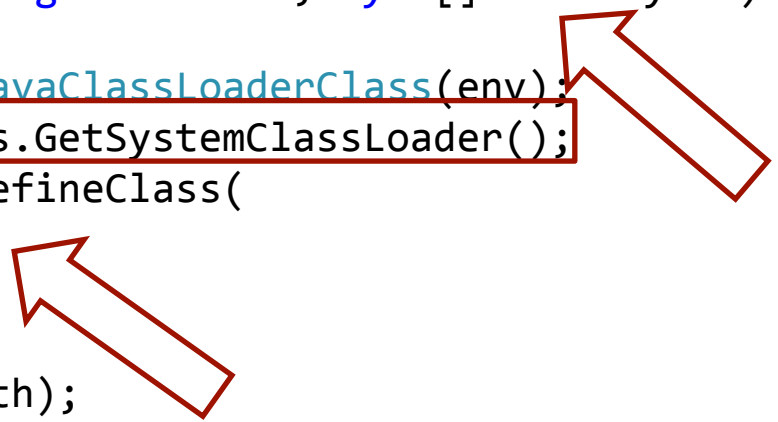
Загрузка классов

- classpath `java -classpath C:\fop\jars ru.kontur.fop.FopService`
- ClassLoader

```
URL[] jars =
    new URL[]{new URL("file:///c:/fop/jars/fop.jar")};
URLClassLoader cl = new URLClassLoader(jars);
Class<?> serviceClass =
    cl.loadClass("ru.kontur.fop.FopService");
```

Загрузка классов

```
public void InjectClass(string className, byte[] classBytes)
{
    var loaderClass = new JavaClassLoaderClass(env);
    var loader = loaderClass.GetSystemClassLoader();
    var loadedClass = env.DefineClass(
            className,
            loader,
            classBytes,
            classBytes.Length);
}
```



Промежуточные итоги

- Запустили виртуальную машину Java внутри .NET процесса
- Реализовали обмен данными между программами на Java и на C#
- Получили готовый работающий прототип решения задачи конвертации документов

Мотивация? Но я же…

- ~~Инфраструктурные ограничения~~
- ~~Перспектива снижения издержек на передачу данных~~

ЕЩЁ «БОЛЕЕ БЛИЗКАЯ» ИНТЕГРАЦИЯ

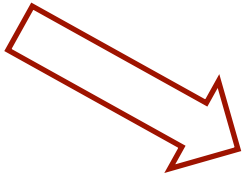
Передача данных через JNI

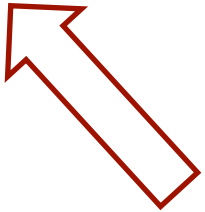
Передача данных через JNI

- JNI: byte[]
- JNI: DirectByteBuffer

JNI: byte[]

```
using (var byteArray = Env.NewByteArray(bytes))
{
    Env.CallObjectMethod(
        obj,
        method,
        new JValue() { PointerValue = byteArray.LocalRef };
}
```





```
public byte[] convert(byte[] data) {
    // ...
}
```

JNI: DirectByteBuffer

```
fixed (byte* b = bytes)
{
    var buffer =
        Env.NewDirectByteBuffer((IntPtr)b, bytes.Length));
    Env.CallIntMethod(
        obj,
        method,
        new JValue() { PointerValue = buffer.LocalRef });
}
```

```
public int convert(ByteBuffer buffer) {
    byte[] data = new byte[buffer.capacity()];
    buffer.get(data);
    // convert
}
```

Ещё тесты

	Среднее, мс	Стд. откл, мс	Относительное время
gRPC	73 065	2 367	100%
JNI array	?	?	?
JNI DBB	?	?	?

Ещё тесты

	Среднее, мс	Стд. откл, мс	Относительное время
gRPC	73 065	2 367	100%
JNI array	56 550	677	77%
JNI DBB	?	?	?

Ещё тесты

	Среднее, мс	Стд. откл, мс	Относительное время
gRPC	73 065	2 367	100%
JNI array	56 550	677	77%
JNI DBB	56 887	867	78%

Утилизация памяти

Performance Counter: сборка мусора

Gen	JSON	gRPC	byte[]	DirectByteBuffer
0	597	227	25	8
1	19	11	3	0
2	1	3	0	0

НАБЛЮДЕНИЯ, ПЛАНЫ, ИТОГИ

Выводы

Где может пригодиться?

- Десктоп – всё в одном процессе (удобно)
- Большой memory traffic (быстро)
- Не нужна дополнительная инфраструктура для работы (просто и надёжно)

Что дальше?

- Загружать Java-библиотеки разных версий
- Проанализировать поведение виртуальных машин .NET и Java в одном процессе
- Понять, куда это богатство ещё применить
- Сделать C#-обёртку удобной

Выводы

- Всегда есть альтернативы
- «Близкая» интеграция .NET и JVM не опасна
- Но технику безопасности никто не отменял
- Как и чтение документации

ЛОВИМ НЕУЛОВИМОГО

Немного хардфана

Ловим неуловимого

```
delegate IntPtr FindClass(
    IntPtr env,
    [MarshalAs(UnmanagedType.LPArray)] byte[] utf);


byte[] utf = Encoding.UTF8.GetBytes(str);
FindClass(env, utf);
```

Нет терминального символа «null» (0x00), но работает!

Значит, «нулевой» байт стоит следом!

Ловим неуловимого

- Так zeroing же!

```
var obj = new object();
```

Ловим неуловимого

- Так zeroing же!
- Так выравнивание же!

```
byte[] bytes = /* ... */
```

Segment	Size, bytes
Sync block	4 (x32), 8 (x64)
Type handle	4 (x32), 8 (x64)
Length	4
Array values	Length
Alignment	0 - 7

Ловим неуловимого

- Так zeroing же!
- Так выравнивание же!
- Так Sync block == 0 же!

```
lock (obj) { /* ... */ }

obj.GetHashCode();
```

Ловим неуловимого

- Так zeroing же!
- Так выравнивание же!
- Так Sync block == 0 же!
- Да руками в память запиши уже!

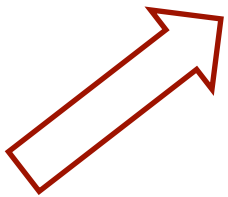
'a'

```
fixed (byte *b = bytes) {
    *(b + bytes.length) = 97;
    /* ... */
}
```

Ловим неуловимого

- Так zeroing же!
- Так выравнивание же!
- Так Sync block == 0 же!
- Да руками в память запиши уже!

```
delegate IntPtr FindClass(
    IntPtr env,
    [MarshalAs(UnmanagedType.LPArray)] byte[] utf);
```



Ловим неуловимого

- Так zeroing же!
- Так выравнивание же!
- Так Sync block == 0 же!
- Да руками в память запиши уже!
- Фикс! Фикс! Фикс!

```
int count = Encoding.UTF8.GetBytesCount(str);
byte[] utf = new byte[count + 1];
Encoding.UTF8.GetBytes(str, 0, str.Length, utf, 0);
FindClass(env, utf);
```

Спасибо за внимание!

ВОПРОСЫ?

СКБ Контур

https://kontur.ru/
http://tech.skbkontur.ru/

Григорий Кошелев

@GregoryKoshelev

kgn@skbkontur.ru

Ссылки

- Apache FOP https://xmlgraphics.apache.org/fop/
- Sharpen https://github.com/mono/sharpen
- IKVM https://www.ikvm.net/
- Protobuf https://developers.google.com/protocol-buffers/
- gRPC http://www.grpc.io/
- JNI http://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/jniTOC.html
- Invocation API http://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/invocation.html
- Mono PInvoke http://www.mono-project.com/docs/advanced/pinvoke/
- Interop Marshalling https://msdn.microsoft.com/ru-ru/library/eaw10et3(v=vs.110).aspx
- Marshal https://msdn.microsoft.com/ru-ru/library/system.runtime.interopservices.marshal(v=vs.110).aspx