Qwertovsky
29 июл 2011 в 14:00

Работа с Java в командной строке

Java*

Туториал Из песочницы

Сейчас уже никто не создает программы в консоли. Используя любимую IDE, разработчик чувствует себя неуютно за чужим компьютером, где её нет. Решив разобраться в работе Ant и Maven, я поймал себя на том, что не смогу собрать приложение без них в консоли.
В данной статье я постарался уместить все этапы проектирования демонстрационного приложения, чтобы не искать справку по каждой команде на просторах Интернета.

От простого к ...

Каждая программа обычно содержится в отдельном каталоге. Я придерживаюсь правила создавать в этом каталоге по крайней мере две папки: src и bin. В первой содержатся исходные коды, во второй — результат компиляции. В данных папках будет структура каталогов, зависящая от пакетов.

Один файл

Можно сделать и без лишних папок.
Берем сам файл HelloWorld.java.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Переходим в каталог, где лежит данный файл, и выполняем команды.

```
javac HelloWorld.java
```

В данной папке появится файл HelloWorld.class. Значит программа скомпилирована. Чтобы запустить

```
java -classpath . HelloWorld
```

Отделяем бинарные файлы от исходников

Теперь сделаем тоже самое, но с каталогами. Создадим каталог HelloWorld и в нем две папки src и bin.
Компилируем

```
javac -d bin src/HelloWorld.java
```

Здесь мы указали, что бинарные файлы будут сохраняться в отдельную папку bin и не путаться с исходниками.

Запускаем

```
java -classpath ./bin HelloWorld
```

Используем пакеты

А то, вдруг, программа перестанет быть просто HelloWorld-ом. Пакетам лучше давать понятное и уникальное имя. Это позволит добавить данную программу в другой проект без конфликта имен. Прочитав некоторые статьи, можно подумать, что для имени пакета обязательно нужен домен. Это не так. Домены — это удобный способ добиться уникальности. Если своего домена нет, воспользуйтесь аккаунтом на сайте (например, ru.habrahabr.mylogin). Он будет уникальным. Учтите, что имена пакетов должны быть в нижнем регистре. И избегайте использования спецсимволов. Проблемы возникают из-за разных платформ и файловых систем.

Поместим наш класс в пакет с именем com.qwertovsky.helloworld. Для этого добавим в начало файла строчку

```
package com.qwertovsky.helloworld;
```

В каталоге src создадим дополнительные каталоги, чтобы путь к файлу выглядел так: src/com/qwertovsky/helloworld/HelloWorld.java.
Компилируем

```
javac -d bin src/com/qwertovsky/helloworld/HelloWorld.java
```

В каталоге bin автоматически создастся структура каталогов как и в src.

```
    HelloWorld
    '---bin
    '   '---com
    '       '---qwertovsky
    '           '---helloworld
    '               '---HelloWorld.class
    '---src
        '---com
            '---qwertovsky
                '---helloworld
                    '---HelloWorld.java
```

Запускаем

```
java -classpath ./bin com.qwertovsky.helloworld.HelloWorld
```

Если в программе несколько файлов

Изменим программу.

HelloWorld.java

```
package com.qwertovsky.helloworld;

public class HelloWorld
{
    public static void main(String[] args)
    {
        int a=2;
        int b=3;
        Calculator calc=new Calculator();
        System.out.println("Hello World!");
        System.out.println(a+"+"+b+"="+calc.sum(a,b));
    }
}
```

Calculator.java

```
package com.qwertovsky.helloworld;

import com.qwertovsky.helloworld.operation.Adder;

public class Calculator
```

```
{
    public int sum(int... a)
    {
        Adder adder=new Adder();
        for(int i:a)
        {
            adder.add(i);
        }
        return adder.getSum();
    }
}
```

Adder.java

```
package com.qwertovsky.helloworld.operation;

public class Adder
{
    private int sum;

    public Adder()
    {
        sum=0;
    }

    public Adder(int a)
    {
        this.sum=a;
    }

    public void add(int b)
    {
        sum+=b;
    }

    public int getSum()
    {
        return sum;
    }
}
```

Компилируем

```
javac  -d bin src/com/qwertovsky/helloworld/HelloWorld.java
    src\com\qwertovsky\helloworld\HelloWorld.java:9: cannot find symbol
```

```
    symbol  : class Calculator
    location: class com.qwertovsky.helloworld.HelloWorld
                Calculator calc=new Calculator();
                ^
    src\com\qwertovsky\helloworld\HelloWorld.java:9: cannot find symbol
    symbol  : class Calculator
    location: class com.qwertovsky.helloworld.HelloWorld
                Calculator calc=new Calculator();
                                ^
    2 errors
```

Ошибка возникла из-за того, что для компиляции нужны файлы с исходными кодами классов, которые используются (класс Calculator). Надо указать компилятору каталог с файлами с помощью ключа -sourcepath.
Компилируем

```
javac -sourcepath ./src -d bin src/com/qwertovsky/helloworld/HelloWorld.java
```

Запускаем

```
java -classpath ./bin com.qwertovsky.helloworld.HelloWorld
    Hello Word
    2+3=5
```

Если удивляет результат

Есть возможность запустить отладчик. Для этого существует jdb.
Сначала компилируем с ключом -g, чтобы у отладчика была информация.

```
javac -g -sourcepath ./src -d bin src/com/qwertovsky/helloworld/HelloWorld.java
```

Запускаем отладчик

```
jdb -classpath bin -sourcepath src com.qwertovsky.helloworld.HelloWorld
    Initializing jdb ...
    >
```

Отладчик запускает свой внутренний терминал для ввода команд. Справку по последним можно вывести с помощью команды help.
Указываем точку прерывания на 9 строке в классе Calculator

```
> stop at com.qwertovsky.helloworld.Calculator:9
    Deferring breakpoint com.qwertovsky.helloworld.Calculator:9.
    It will be set after the class is loaded.
```

Запускаем на выполнение.

```
> run
    run com.qwertovsky.helloworld.HelloWorld
    Set uncaught java.lang.Throwable
    Set deferred uncaught java.lang.Throwable
    >
    VM Started: Set deferred breakpoint com.qwertovsky.helloworld.Calculator:9
    Hello World!

    Breakpoint hit: "thread=main", com.qwertovsky.helloworld.Calculator.sum(), line
    9               Adder adder=new Adder();
```

Чтобы сориентироваться можно вывести кусок исходного кода, где в данный момент находится курссор.

```
main[1] list
    5    public class Calculator
    6    {
    7       public int sum(int... a)
    8       {
    9 =>            Adder adder=new Adder();
    10              for(int i:a)
    11              {
    12                      adder.add(i);
    13              }
    14              return adder.getSum();
```

Узнаем, что из себя представляет переменная a.

```
main[1] print a
     a = instance of int[2] (id=340)
main[1] dump a
     a = {
    2, 3
    }
main[1] stop at com.qwertovsky.helloworld.operation.Adder:19
```

```
    Deferring breakpoint com.qwertovsky.helloworld.operation.Adder:19.
    It will be set after the class is loaded.
```

Продолжим исполнение.

```
main[1] cont
    > Set deferred breakpoint com.qwertovsky.helloworld.operation.Adder:19

    Breakpoint hit: "thread=main", com.qwertovsky.helloworld.operation.Adder.add(),
    19              sum+=b;

main[1] list
    15      }
    16
    17      public void add(int b)
    18      {
    19 =>           sum+=b;
    20      }
    21
    22      public int getSum()
    23      {
    24              return sum;
main[1] print sum
     sum = 0
main[1] print b
     b = 2
```

Выполним код в текущей строке и увидим, что sum стала равняться 2.

```
main[1] step
    >
    Step completed: "thread=main", com.qwertovsky.helloworld.operation.Adder.add(),
    20      }

main[1] print sum
     sum = 2
```

Поднимемся из класса Adder в вызвавший его класс Calculator.

```
main[1] step up
    >
```

```
    Step completed: "thread=main", com.qwertovsky.helloworld.Calculator.sum(), line
    10              for(int i:a)
```

Удаляем точку прерывания

```
main[1] clear com.qwertovsky.helloworld.operation.Adder:19
    Removed: breakpoint com.qwertovsky.helloworld.operation.Adder:19
main[1] step
    >
    Step completed: "thread=main", com.qwertovsky.helloworld.Calculator.sum(), line
    12                      adder.add(i);
```

Можно избежать захода в методы, используя команду next.

```
main[1] next
    >
    Step completed: "thread=main", com.qwertovsky.helloworld.Calculator.sum(), line
    10              for(int i:a)

main[1] next
    >
    Step completed: "thread=main", com.qwertovsky.helloworld.Calculator.sum(), line
    14              return adder.getSum();
```

Проверяем значение выражения и завершаем выполнение.

```
main[1] eval adder.getSum()
     adder.getSum() = 5
main[1] cont
    > 2+3=5

    The application exited
```

Хорошо бы протестировать

Используем JUnit.

```
package com.qwertovsky.helloworld;
```

```
import static org.junit.Assert.*;

import java.util.Arrays;
import java.util.Collection;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized.Parameters;

@RunWith(value=org.junit.runners.Parameterized.class)
public class TestCalculator
{
    int expected;
    int[] arg;

    @Parameters
    public static Collection<int[][]> parameters()
    {

        return Arrays.asList(new int[][][]{
                {{4}, {2, 2}}
                ,{{-1},{4, -5}}
                ,{{0},{0,0,0}}
                ,{{0},{}}
                });
    }

    public TestCalculator(int[] expected, int[] arg)
    {
        this.expected=expected[0];
        this.arg=arg;
    }

    @Test
    public void testSum()
    {
        Calculator c=new Calculator();
        assertEquals(expected,c.sum(arg));
    }
}
```

Компилируем

```
mkdir test_bin
```

```
javac  -classpath lib/path/junit-4.8.2.jar  -sourcepath ./src -d test_bin test/com/
```

Запускаем. В качестве разделителя нескольких путей в classpath в Windows используется ';', в Linux — ':'. В консоли Cygwin не работают оба разделителя. Возможно, должен работать ';', но он воспринимается как разделитель команд.

```
java  -classpath lib/path/junit-4.8.2.jar:./test_bin  org.junit.runner.JUnitCore co
    JUnit version 4.8.2
    ....
    Time: 0,031

    OK (4 tests)
```

Создадим библиотеку

Класс Calculator оказался полезным и может быть использован во многих проектах. Перенесем всё, что касается класса Calculator в отдельный проект.

```
    HelloWorld
    '---bin
    '---src
        '---com
            '---qwertovsky
                '---helloworld
                    '---HelloWorld.java
    Calculator
    '---bin
    '---src
    '   '---com
    '       '---qwertovsky
    '           '---calculator
    '               '---Calculator.java
    '               '---operation
    '                   '---Adder.java
    '---test
        '---com
            '---qwertovsky
                '---calculator
                    '---TestCalculator.java
```

Измените также назавания пакетов в исходных текстах. В HelloWorld.java нужно будет добавить строку

```
import com.qwertovsky.calculator.Calculator;
```

Компилируем.

```
cd Calculator
javac -sourcepath src -d bin src/com/qwertovsky/calculator/Calculator.java
```

Делаем архив jar

```
jar cvf calculator.jar -C bin .
    added manifest
    adding: com/(in = 0) (out= 0)(stored 0%)
    adding: com/qwertovsky/(in = 0) (out= 0)(stored 0%)
    adding: com/qwertovsky/calculator/(in = 0) (out= 0)(stored 0%)
    adding: com/qwertovsky/calculator/Calculator.class(in = 497) (out= 373)(deflate
    adding: com/qwertovsky/calculator/operation/(in = 0) (out= 0)(stored 0%)
    adding: com/qwertovsky/calculator/operation/Adder.class(in = 441) (out= 299)(de
```

С помощью ключа -C мы запустили программу в каталоге bin.

Надо узнать, что у библиотеки внутри

Можно распаковать архив zip-распаковщиком и посмотреть, какие классы есть в библиотеке. Информацию о любом классе можно получить с помощью дизассемблера javap.

```
javap -c -classpath calculator.jar com.qwertovsky.calculator.Calculator
    Compiled from "Calculator.java"
    public class com.qwertovsky.calculator.Calculator extends java.lang.Object{
    public com.qwertovsky.calculator.Calculator();
      Code:
       0:   aload_0
       1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
       4:   return

    public int sum(int[]);
      Code:
       0:   new #2; //class com/qwertovsky/calculator/operation/Adder
       3:   dup
```

```
       4:   invokespecial   #3; //Method com/qwertovsky/calculator/operation/Adder.
       7:   astore_2
       8:   aload_1
       9:   astore_3
       10:  aload_3
       11:  arraylength
       12:  istore  4
       14:  iconst_0
       15:  istore  5
       17:  iload   5
       19:  iload   4
       21:  if_icmpge   42
       24:  aload_3
       25:  iload   5
       27:  iaload
       28:  istore  6
       30:  aload_2
       31:  iload   6
       33:  invokevirtual   #4; //Method com/qwertovsky/calculator/operation/Adder.
       36:  iinc    5, 1
       39:  goto    17
       42:  aload_2
       43:  invokevirtual   #5; //Method com/qwertovsky/calculator/operation/Adder.
       46:  ireturn

    }
```

Из результата видно, что класс содержит кроме пустого конструктора, ещё один метод sum, внутри которого в цикле вызывается метод add класса Adder. По завершении метода sum, вызывается Adder.getSum().
Без ключа -c программа выдаст только список переменных и методов (если использовать -private, то всех).

```
javap -private  -classpath calculator.jar com.qwertovsky.calculator.operation.Adder
    Compiled from "Adder.java"
    public class com.qwertovsky.calculator.operation.Adder extends java.lang.Object
        private int sum;
        public com.qwertovsky.calculator.operation.Adder();
        public com.qwertovsky.calculator.operation.Adder(int);
        public void add(int);
        public int getSum();
    }
```

Лучше снабдить библиотеку документацией

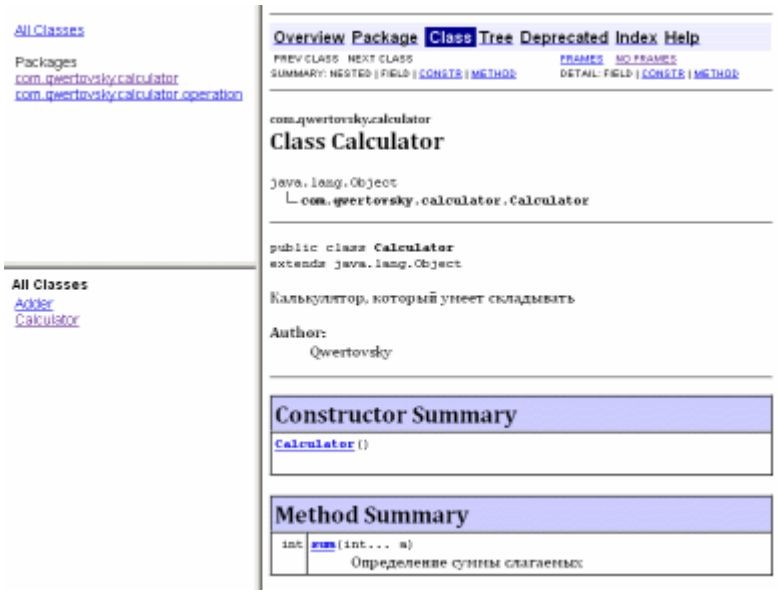
Изменим для этого класс калькулятора.

```
package com.qwertovsky.calculator;

import com.qwertovsky.calculator.operation.Adder;

/**
 * Калькулятор, который умеет складывать
 * @author Qwertovsky
 *
 */
public class Calculator
{
    /**
     * Определение суммы слагаемых
     * @param a массив слагаемых
     * @return сумма
     */
    public int sum(int... a)
    {
        Adder adder=new Adder();
        for(int i:a)
        {
            adder.add(i);
        }
        return adder.getSum();

    }
}
```

Документацию можно создать следующей командой. При ошибке программа выдаст список возможных опций.

```
mkdir doc
javadoc -d doc -charset utf-8  -sourcepath src -author -subpackages com.qwertovsky.
```

В результате получиться следующее



Можно подписать jar-архив

Если требуется подписать свою библиотеку цифровой подписью, на помощь придут keytool и jarsigner.
Генерируем подпись.

```
keytool -genkey -keyalg rsa -keysize 2048 -alias qwertokey -keystore path/to/qwerto
    Enter keystore password:
    Re-enter new password:
    What is your first and last name?
      [Unknown]:  Valery Qwertovsky
    What is the name of your organizational unit?
      [Unknown]:  Qwertovsky
    What is the name of your organization?
      [Unknown]:  Qwertovsky
    What is the name of your City or Locality?
      [Unknown]:  Tver
    What is the name of your State or Province?
      [Unknown]:  Tverskaya obl.
    What is the two-letter country code for this unit?
      [Unknown]:  RU
    Is CN=Valery Qwertovsky, OU=Qwertovsky, O=Qwertovsky, L=Tver, ST=Tverskaya  obl
      [no]:  y

    Enter key password for <qwertokey>
            (RETURN if same as keystore password):
    Re-enter new password:
```

Генерируем Certificate Signing Request (CSR)

```
keytool -certreq -file path/to/qwertokey.crt -alias qwertokey -keystore path/to/qwe
```

Содержимое полученного файла отправляем в центр сертификации. От центра сертификации получаем сертификат. Сохраняем его в файле (например, qwertokey.cer) и импортируем в хранилище

```
keytool -import -trustcacerts -keystore path/to/qwert.keystore -alias qwertokey -fi
```

Подписываем jar-архив

```
jarsigner -keystore path/to/qwerto.keystore calculator.jar qwertokey
```

Файл qwertokey.cer отправляем всем, кто хочет проверить архив. Проверяется он так

```
jarsigner -verify -verbose -certs -keystore path/to/qwerto.keystore calculator.jar
```

Использование библиотеки

Есть программа HelloWorld, которая использует библиотечный класс Calculator. Чтобы скомпилировать и запустить программу, нужно присоединить библиотеку.
Компилируем

```
cd HelloWorld
javac -sourcepath src -d bin -classpath path/to/calculator.jar src/com/qwertovsky/h
```

Запускаем

```
java -classpath bin:path/to/calculator.jar com.qwertovsky.helloworld.HelloWorld
```

Собираем программу

Это можно сделать по-разному.

Первый способ

```
cd HelloWorld
echo main-class: com.qwertovsky.helloworld.HelloWorld>manifest.mf
echo class-path: lib/calculator.jar >>manifest.mf
mkdir lib
cp path/to/calculator.jar lib/calculator.jar
jar -cmf manifest.mf helloworld.jar  -C bin .
```

Здесь есть тонкости.
В строке

```
main-class: com.qwertovsky.helloworld.HelloWorld
```

не должно быть пробелов в конце.
Вторая тонкость описана в [3]: в этой же строке должен стоять перенос на следующую строку. Это если манифест помещается в архив сторонним архиватором.
Программа jar не включит в манифест последнюю строку из манифеста, если в конце не стоит перенос строки.
Ещё момент: в манифесте не должно быть пустых строк между строками. Будет выдана ошибка «java.io.IOException: invalid manifest format».

При использовании команды echo надо следить только за пробелом в конце строки с main-class.

Второй способ

```
cd HelloWorld
echo class-path: lib/calculator.jar >manifest.mf
mkdir lib
cp path/to/calculator.jar lib/calculator.jar
jar -cmef manifest.mf com.qwertovsky.helloworld.HelloWorld  helloworld.jar  -C bin
```

В данном способе избегаем ошибки с пробелом в main-class.

Третий способ

```
cd HelloWorld
mkdir lib
cd lib
```

```
jar -xvf path/to/calculator.jar com/
      created: com/
      created: com/qwertovsky/
      created: com/qwertovsky/calculator/
     inflated: com/qwertovsky/calculator/Calculator.class
      created: com/qwertovsky/calculator/operation/
     inflated: com/qwertovsky/calculator/operation/Adder.class

cd ..
cp  -r bin/* lib/
jar -cef com.qwertovsky.helloworld.HelloWorld  helloworld.jar  -C lib .
rm -r lib
```

Включили код нужной библиотеки в исполняемый файл.

Запуск исполняемого jar-файла

Файл calculator.jar исполняемым не является. А вот helloworld.jar можно запустить.
Если архив был создан первыми двумя способами, то рядом с ним в одном каталоге должна находится папка lib с файлом calculator.jar. Такие ограничения из-за того, что в манифесте в class-path указан путь относительно исполняемого файла.

```
cd Calculator
ls ../HelloWorld/lib
    calculator.jar
java -jar ../HelloWorld/helloworld.jar
```

При использовании третьего способа нужные библиотеки включаются в исполняемый файл. Держать рядом нужные библиотеки не требуется. Запускается аналогично.

```
java -jar ../HelloWorld/helloworld.jar
```

Как быть с приложениями JavaEE

Аналогично. Только библиотеки для компиляции нужно брать у сервера приложений, который используется. Если я использую JBoss, то для компиляции сервлета мне нужно будет выполнить примерно следующее

```
javac -classpath path/to/jboss/common/lib/jboss-servlet*.jar  -d ./classes src/com/
```

Структура архива JavaEE-приложения должна соответствовать определенному формату. Например

```
my.ear
`---META-INF
|   `---manifest.mf
`---lib
|   `---mylib.jar
`---my.war
|   `---META-INF
|   |   `---manifest.mf
|   `---WEB-INF
|   |   `---lib
|   |   |   `---myweblib.jar
|   |   `---classes
|   |   |   `---com
|   |   |       `---...
|   |   `---web.xml
|   `---index.html
|   `---<остальное веб-содержимое (страницы, изображения)>
`---myejb.jar
```

Способы запуска приложения на самом сервере с помощью командной строки для каждого сервера различны.

Надеюсь, данная статья станет для кого-нибудь шпаргалкой для работы с Java в командной строке. Данные навыки помогут понять содержание и смысл Ant-скриптов и ответить на собеседовании на более каверзные вопросы, чем «Какая IDE Вам больше нравится?».

Ещё почитать

1. Elliotte Rusty Harold. «Рекомендации по управлению classpath в UNIX и Mac OS X»
2. Elliotte Rusty Harold. «Рекомендации по управлению classpath в Windows»
3. Евгений Матюшкин aka Skipy. «Ликбез»
4. Lesson: Packaging Programs in JAR Files
5. Brian Goetz. «Теория и практика Java: Мне нужно задокументировать ЭТО?»
6. Евгений Матюшкин aka Skipy. «Создание собственных тегов javadoc»
7. Создание и использование архивов Java
8. Sun Java Signing
9. javac — Java programming language compiler
10. java — the Java application launcher
11. jdb — The Java Debugger

12. javap — The Java Class File Disassembler
13. javadoc — The Java API Documentation Generator
14. jarsigner — JAR Signing and Verification Tool
15. jar — The Java Archive Tool
16. keytool — Key and Certificate Management Tool

Теги: java, командная строка, консоль, разработка

Хабы: Java

50 Карма 0 Рейтинг

Валерий Квертовский @Qwertovsky

Full-Stack Software Engineer

Подписаться

Комментарии 25

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

inetstar
23 часа назад

Решаем загадку Джиндоша из Dishonored 2 на SQL перебором с возвратом

Средний 17 мин 6.1K

Кейс

+57 38 22

stalkermustang
7 часов назад

Большие и чёрные (ящики): что мы знаем о том, как «думают» нейросети?

Средний 30 мин 10K

Обзор

+56 51 25

mikeveng73
17 часов назад

pico-xt — старая добрая PC/XT на Мурмуляторе

Простой 2 мин 2.2K

Обзор

+27 13 29

frog
12 часов назад

Ещё слово о процедурной графике

5 мин 1.6K

+22 10 2

tech_priestess
16 часов назад

Размерность Минковского и Two Nearest Neighbours (TwoNN)

Сложный 7 мин 2.3K

Туториал

+20 21 17

mikeveng73
13 часов назад

pico-bk — БК0010/11М на Мурмуляторе

Простой 2 мин 1.4K

Мнение

+18 9 6

spring_aio
23 часа назад

Структурное логирование в Spring Boot 3.4

Простой 5 мин 3.2K

Обзор Перевод

+17 48 1

Baron_Kir
4 часа назад

Деннис Макалистэйр Ритчи. Между Unix и C

Простой 18 мин 749

Ретроспектива

+14 5 0

it_union
6 часов назад

Разработчики Greedfall 2 таки раскачали лодку

2 мин 1.5K

+13 0 0

veseluha
21 час назад

Раскрываем секреты роя: оптимизация на Python с помощью PSO

Средний 13 мин 2.5K

Туториал Перевод

+13 51 10

Показать еще

ВОПРОСЫ И ОТВЕТЫ

Разница между root exception и cause exception в Spring Framework?

Java · Простой · 0 ответов

Как реализовать выбор и подтягивание данных на форму?

Java · Средний · 0 ответов

Возможно ли написать так сказать кейлоггер на java если да то что использовать?

Java · Простой · 2 ответа

Как вычитать код страницы в Java?

JavaScript · Простой · 3 ответа

Как сделать группировку в PostgreSQL по столбцам из списка выборки, выражениям и без GROUP BY?

PostgreSQL · Простой · 1 ответ

Больше вопросов на Хабр Q&A

ЧИТАЮТ СЕЙЧАС

МС-21: инженерный триумф российского авиастроения

84K 463

Каждый день хочу завязать с бетоном, но 45 млн в год не отпускают

100K 72

Какой роутер для OpenWrt купить в 2025 году?

33K 129

Большие и чёрные (ящики): что мы знаем о том, как «думают» нейросети?

10K 25

В России выпустили модуль для апгрейда телефона до версии с eSIM

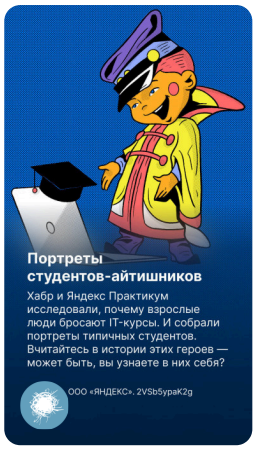
3.4K 48

ИСТОРИИ

Добро пожаловать на борт!

Архитектор и сеньорская борода

Лето. Отпуск. Мысли о будущем



Выгорающие IT-студенты: какие они?

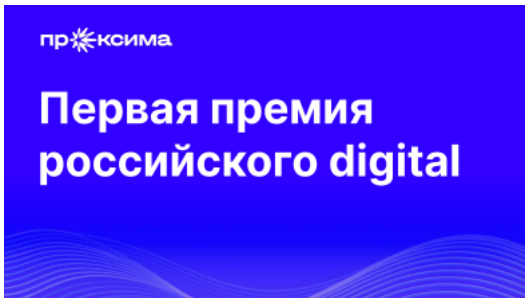
Как рассказать о событии на Хабре

РАБОТА

Java разработчик
326 вакансий

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ



27 августа – 7 октября

Премия digital-кейсов «Проксима»

Москва • Онлайн

Маркетинг Другое

Больше событий в календаре

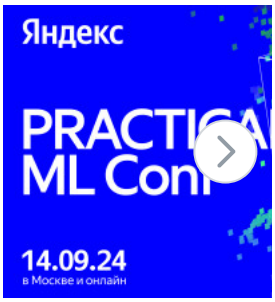
13 сентября

Премия для стартапов Startech.Awards

Москва

Менеджмент Другое

Больше событий в календаре



14 сентября

Конференция Conf

Москва • Онлайн

Разработка

Больше событий в к