



# Почему вы не можете не использовать API

Нужное





# **Что такое API в широком смысле**



# Что такое API

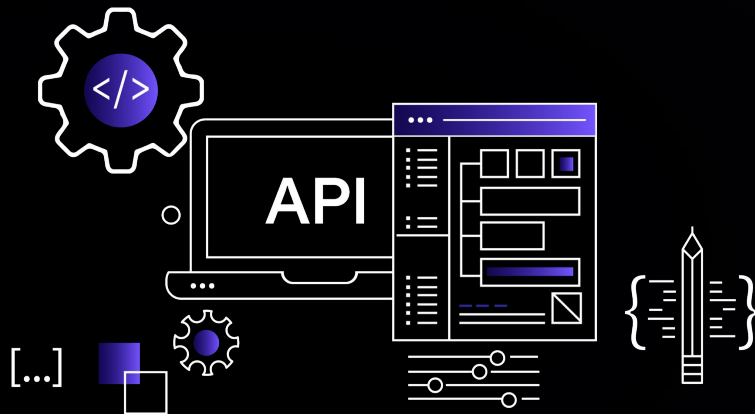
API (Application programming interface) — это контракт, который предоставляет программа. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».

[habr](#)



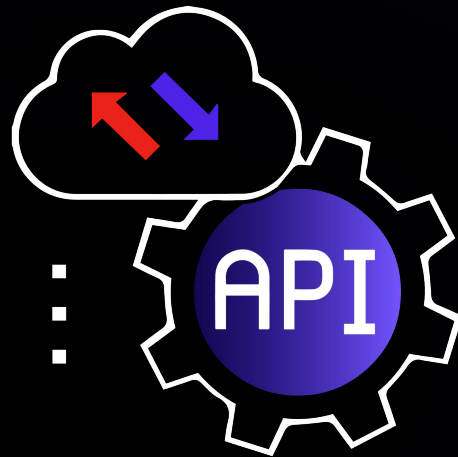
# Что такое API для нас

Это возможности, который предоставляют разработчики языка для удобного взаимодействия с его функционалом



# Что такое API для нас на примерах

1. Строки
2. Работа с файловой системой
3. Логирование
4. Импорт
5. Xml



**Строки**



# Что такое API для нас: строки

Простой пример

Создать строку из 1 млн плюсиков. Как?

```
String str = "";  
for (int i = 0; i < 1_000_000; i++) {  
    str += "+";  
}
```



# Что такое API для нас: строки

Простой пример

Создать строку из 1 млн плюсиков. Как?

```
String str = "";  
for (int i = 0; i < 1_000_000; i++) {  
    str += "+";  
}
```

```
// ≈41000 ms
```





# Что такое API для нас: строки

Простой пример

Создать строку из 1 млн плюсиков. Как?

```
StringBuilder sb = new StringBuilder();  
for (int i = 0; i < 1_000_000; i++) {  
    sb.append("+");  
}
```



# Что такое API для нас: строки

Простой пример

Создать строку из 1 млн плюсиков. Как?

```
StringBuilder sb = new StringBuilder();  
for (int i = 0; i < 1_000_000; i++) {  
    sb.append("+");  
}
```

```
// ≈9 ms
```



# Что такое API для нас: строки

## Весь код

```
public class program {  
    public static void main(String[] args) {  
        var s = System.currentTimeMillis();  
        //String str = "";  
        StringBuilder sb = new StringBuilder();  
        for (int i = 0; i < 1_000_000; i++) {  
            //str += "+";  
            sb.append("+");  
        }  
        System.out.println(System.currentTimeMillis() - s);  
        //System.out.println(str);  
        //System.out.println(sb);  
    }  
}
```



# Что такое API для нас: строки

**concat():** объединение строк

**valueOf():** преобразует Object в строковое представление (завязан на toString())

**join():** объединяет набор строк в одну с учетом разделителя

**charAt():** получение символа по индексу

**indexOf():** первый индекс вхождения подстроки

**lastIndexOf():** последний индекс вхождения подстроки

**startsWith()/endsWith():** определяет, начинается/заканчивается ли строка с подстроки

**replace():** замена одной подстроки на другую



# Что такое API для нас: строки

**trim():** удаляет начальные и конечные пробелы

**substring():** возвращает подстроку, см. аргументы

**toLowerCase()/toUpperCase():** возвращает новую строку в нижнем/верхнем регистре

**compareTo():** сравнивает две строки

**equals():** сравнивает строки с учетом регистра

**equalsIgnoreCase():** сравнивает строки без учета регистра

**regionMatches():** сравнивает подстроки в строках



# Что такое API для нас: строки

```
public class program {  
    public static void main(String[] args) {  
        String[] name = { "С", "е", "р", "г", "е", "й" };  
        String sk = "СЕРГЕЙ КА.";  
        System.out.println(sk.toLowerCase()); // сергей ка.  
        System.out.println(String.join("", name)); // Сергей  
        System.out.println(String.join("", "С", "е", "р", "г", "е", "й"));  
        // С,е,р,г,е,й  
        System.out.println(String.join(", ", "С", "е", "р", "г", "е", "й"));  
    }  
}
```



Что такое API для нас: строки

# String **VS** StringBuilder



# Что такое API для нас: строки

Много изменений – **String**





# Что такое API для нас: строки

Много изменений – **String**

Много преобразований – **StringBuilder**



# Работа с файловой системой



Работа с файловой системой

**Сколько разного в файловой  
системе?**



Работа с файловой системой

**Сколько разного в файловой  
системе?**

**Каталоги и файлы**



# Работа с файловой системой. Файлы

Для работы нужно:

`File <имя> = new File(<полный путь к файлу>);`

```
File f1 = new File("file.txt");
```

```
File f2 = new File("/Users/sk/vscode/java_projects/file.txt");
```

Что предпочтительнее?



# Работа с файловой системой. Файлы

```
import java.io.File;

public class fileSystemDemo {
    public static void main(String[] args) {
        String pathProject = System.getProperty("user.dir");
        String pathFile = pathProject.concat("/file.txt");
        File f3 = new File(pathFile);
        System.out.println(f3.getAbsolutePath());
        // /Users/sk/vscode/java_projects/file.txt
        // C:/Users/Sk/Documents/xxx/brainexplosion/java/file.txt
    }
}
```



# Работа с файловой системой. Файлы

**Ошибка на ошибке  
и ошибкой погоняет**

**Как быть?**



# Работа с файловой системой. Файлы. Ошибки





# Работа с файловой системой. Файлы. Ошибки

```
try {  
    Код, в котором может появиться ошибка  
} catch (Exception e) {  
    Обработка, если ошибка случилась  
}  
finally {  
    Код, который выполнится в любом случае  
}
```



# Работа с файловой системой. Файлы. Ошибки

```
import java.io.File;
public class tryDemo {
    public static void main(String[] args) {
        try {
            String pathProject = System.getProperty("user.dir");
            String pathFile = pathProject.concat("/file.txt");
            File f3 = new File(pathFile);
            System.out.println("try");
        } catch (Exception e) {
            System.out.println("catch");
        }
        finally
        { System.out.println("finally"); }
    }
}
```



# Работа с файловой системой. Файлы. Ошибки

```
import java.io.File;
public class tryDemo {
    public static void main(String[] args) {
        try {
            String pathProject = System.getProperty("user.dir");
            String pathFile = pathProject.concat("/file.txt");
            File f3 = new File(pathFile);
            System.out.println("try");
        } catch (Exception e) {
            System.out.println("catch");
        }
        finally
        { System.out.println("finally"); }
    }
}
```

**Всё ли ок?**



# Работа с файловой системой. Файлы. Ошибки

```
import java.io.File;
public class tryDemo {
    public static void main(String[] args) {
        try {
            String pathProject = System.getProperty("user.dir");
            String pathFile = pathProject.concat("/file.txt");
            File file = new File(pathFile);
            if (file.createNewFile()) {
                System.out.println("file.created");
            }
            else {
                System.out.println("file.existed");
            }
        } catch (Exception e) {
            System.out.println("catch");
        } finally {
            System.out.println("finally");
        }
    }
}
```



# Работа с файловой системой. Файлы. Ошибки

```
String line = "empty";
try {
    File file = new File(pathFile);
    if (file.createNewFile()) {
        System.out.println("file.created"); }
    else {
        BufferedReader bufReader =
            new BufferedReader(new FileReader(file));
        System.out.println("file.existed");
        line = bufReader.readLine();
        bufReader.close(); }
} catch (Exception e) {
    //e.printStackTrace();
} finally {
    System.out.println(line);
}
```



# Работа с файловой системой

**isHidden():** возвращает истину, если каталог или файл является скрытым

**length():** возвращает размер файла в байтах

**lastModified():** возвращает время последнего изменения файла или каталога

**list():** возвращает массив файлов и подкаталогов, которые находятся в каталоге

**listFiles():** возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

**mkdir():** создает новый каталог

**renameTo(File dest):** переименовывает файл или каталог



# Работа с файловой системой

**length():** возвращает размер файла в байтах

**lastModified():** возвращает время последнего изменения файла или каталога

**list():** возвращает массив файлов и подкаталогов, которые находятся в каталоге

**listFiles():** возвращает массив файлов и подкаталогов, которые находятся в определенном каталоге

**mkdir():** создает новый каталог

**renameTo(File dest):** переименовывает файл или каталог



# Работа с файловой системой. Каталоги

```
import java.io.File;
public class Ex0043 {
    public static void main(String[] args) {
        String pathProject = System.getProperty("user.dir");
        String pathDir = pathProject.concat("/files");
        File dir = new File(pathDir);
        System.out.println(dir.getAbsolutePath());
        if (dir.mkdir()) {
            System.out.println("+");
        } else {
            System.out.println("-");
        }
        for (String fname : dir.list()) {
            System.out.println(fname);
        }
    }
}
```





# Работа с файловой системой

Бинарные файлы – что это?

**Демонстрация**



# Логирование



# Логирование

Логи содержат системную информацию работы программного или аппаратного комплекса.

В них записываются действия разного приоритета от пользователя, или программного продукта.

Процесс ведения логов называют “логированием” (журналированием).



# Логирование. Использование

```
Feb 14 1994 05:08:33 WATCHDOG: [FAUPGRADE][auto_firmware_check:(7285)]no need to upgrade firmware
Feb 14 1994 07:03:41 ntp: start NTP update
Feb 14 1994 10:14:02 syslog: wlceventd_proc_event(527): eth1: Auth BC:DD:C2:88:E2:3F, status: Successful (0), rssi:0
Feb 14 1994 10:14:02 syslog: wlceventd_proc_event(556): eth1: Assoc BC:DD:C2:88:E2:3F, status: Successful (0), rssi:0
Feb 14 1994 10:20:20 syslog: wlceventd_proc_event(527): eth1: Auth D4:A6:51:07:54:BB, status: Successful (0), rssi:0
Feb 14 1994 10:20:20 syslog: wlceventd_proc_event(556): eth1: Assoc D4:A6:51:07:54:BB, status: Successful (0), rssi:0
Feb 14 1994 10:20:20 syslog: wlceventd_proc_event(527): eth1: Auth D4:A6:51:01:F4:E9, status: Successful (0), rssi:0
Feb 14 1994 10:20:20 syslog: wlceventd_proc_event(556): eth1: Assoc D4:A6:51:01:F4:E9, status: Successful (0), rssi:0
Feb 14 1994 11:39:08 syslog: wlceventd_proc_event(491): eth1: Deauth_ind 48:3F:DA:91:19:E0, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:03:06 syslog: wlceventd_proc_event(491): eth2: Deauth_ind 18:3E:EF:E9:DF:15, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:03:08 syslog: wlceventd_proc_event(491): eth1: Deauth_ind BC:DD:C2:88:E2:3F, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:03:19 syslog: wlceventd_proc_event(491): eth2: Deauth_ind C2:D4:93:FB:9F:FB, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:03:31 syslog: wlceventd_proc_event(491): eth2: Deauth_ind FE:1F:80:6E:4B:09, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:06:34 syslog: wlceventd_proc_event(491): eth1: Deauth_ind D4:A6:51:01:F4:E9, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 13:06:34 syslog: wlceventd_proc_event(491): eth1: Deauth_ind D4:A6:51:07:54:BB, status: 0, reason: Disassociated due to inactivity (4), rssi:0
Feb 14 1994 18:07:09 syslog: wlceventd_proc_event(527): eth1: Auth 72:34:C1:43:EC:16, status: Successful (0), rssi:0
Feb 14 1994 18:07:09 syslog: wlceventd_proc_event(527): eth1: Auth 72:34:C1:43:EC:16, status: Successful (0), rssi:0
Feb 14 1994 18:07:09 syslog: wlceventd_proc_event(556): eth1: Assoc 72:34:C1:43:EC:16, status: Successful (0), rssi:0
Feb 14 1994 18:07:23 syslog: wlceventd_proc_event(527): eth1: Auth FA:7F:A4:CD:2F:4B, status: Successful (0), rssi:0
Feb 14 1994 18:07:23 syslog: wlceventd_proc_event(556): eth1: Assoc FA:7F:A4:CD:2F:4B, status: Successful (0), rssi:0
Feb 14 1994 18:08:10 syslog: wlceventd_proc_event(527): eth1: Auth 84:CC:A8:86:02:33, status: Successful (0), rssi:0
Feb 14 1994 18:08:10 syslog: wlceventd_proc_event(556): eth1: Assoc 84:CC:A8:86:02:33, status: Successful (0), rssi:0
Feb 14 1994 18:08:15 syslog: wlceventd_proc_event(527): eth1: Auth BC:DD:C2:88:E2:3F, status: Successful (0), rssi:0
Feb 14 1994 18:08:15 syslog: wlceventd_proc_event(556): eth1: Assoc BC:DD:C2:88:E2:3F, status: Successful (0), rssi:0
Feb 14 1994 18:09:14 syslog: wlceventd_proc_event(527): eth2: Auth 18:3E:EF:E9:DF:15, status: Successful (0), rssi:0
Feb 14 1994 18:09:14 syslog: wlceventd_proc_event(556): eth2: Assoc 18:3E:EF:E9:DF:15, status: Successful (0), rssi:0
```



# Логирование. Использование. Основы

## Использование

```
Logger logger = Logger.getLogger()
```

## Уровни важности

INFO, DEBUG, ERROR, WARNING и др.

## Вывод

```
ConsoleHandler info = new ConsoleHandler();  
log.addHandler(info);
```

Формат вывода: структурированный, абы как\*

```
XMLFormatter, SimpleFormatter
```



# Логирование. Использование. Основы

```
import java.util.logging.*;
public class Ex0043 {
    public static void main(String[] args) {
        Logger logger = Logger.getLogger(Ex0043.class.getName());
        logger.setLevel(Level.INFO);
        ConsoleHandler ch = new ConsoleHandler();
        logger.addHandler(ch);
        SimpleFormatter sFormat = new SimpleFormatter();
        ch.setFormatter(sFormat);
        logger.log(Level.WARNING, "Тестовое логирование");
        logger.info("Тестовое логирование");
    }
}
```



# Логирование. Использование. Основы

```
import java.util.logging.*;
public class Ex0043 {
    public static void main(String[] args) {
        Logger logger = Logger.getLogger(Ex0043.class.getName());
        logger.setLevel(Level.INFO);
        ConsoleHandler ch = new ConsoleHandler();
        logger.addHandler(ch);
        //SimpleFormatter sFormat = new SimpleFormatter();
        XMLFormatter xml = new XMLFormatter();
        //ch.setFormatter(sFormat);
        ch.setFormatter(xml);
        logger.log(Level.WARNING, "Тестовое логирование");
        logger.info("Тестовое логирование");
    }
}
```



**Пример**





# Демонстрация


Написать программу для работы с бинарными файлами.  
Предусмотреть логирование всех действий.





**ИТОГИ**  
**Здесь вам не тут!**



**Спасибо**   
**за внимание**

