

Professor Hans Noodles

41 уровень

24.05.2019   24818   31

## Зачем нужно логирование

Статья из группы **Java Developer**  
43181 участник

Вы в группе

Привет!

При написании лекций я особо отмечаю, если какая-то конкретная тема обязательно будет использоваться в реальной работе.



Так вот, ВНИМАНИЕ!

Тема, которой мы коснемся сегодня, точно пригодится тебе на всех твоих проектах с первого дня работы.

Мы поговорим о логировании.

Тема эта совсем не сложная (я бы даже сказал легкая). Но на первой работе и без того будет достаточно стресса, чтобы еще разбираться с очевидными вещами, поэтому лучше досконально разобрать ее сейчас :)

Итак, начнем.

Что такое логирование?

**Логирование — это запись куда-то данных о работе программы.** Место, куда эти данные записываются называется «лог».

НАЧАТЬ ОБУЧЕНИЕ

Начнем с «куда».

Записывать данные о работе программы можно во множество разных мест. Например, ты во время учебы часто выводил данные в консоль с помощью `System.out.println()`. Это настоящее логирование, хоть и самое простое.

Конечно, для клиента или команды поддержки продукта это не очень удобно: они явно не захотят устанавливать IDE и мониторить консоль :)

Есть и более привычный человеку формат записи информации — в текстовый файл. Людям гораздо удобнее читать их в таком виде, и уж точно гораздо удобнее хранить!

Теперь второй вопрос: **какие данные о работе программы должны записываться в лог?**

А вот здесь все зависит от тебя!

Система логирования в Java очень гибкая. Ты можешь настроить ее таким образом, что в лог попадет весь ход работы твоей программы.

Это, с одной стороны, хорошо. Но с другой — представь себе, каких размеров могут достичь логи Facebook или Twitter, если туда писать вообще все.

У таких крупных компаний наверняка есть возможность хранить даже такое количество информации. Но вообрази, как сложно будет искать информацию об одной критической ошибке в логах на 500 гигабайт текста?

Это даже хуже, чем иголка в стоге сена. Поэтому логирование в Java можно настроить так, чтобы в журнал (лог) записывались только данные об ошибках. Или даже только о критических ошибках!

Хотя, говорить «логирование в Java» не совсем верно.

Дело в том, что потребность ведения логов возникла у программистов раньше, чем этот функционал был добавлен в язык.

И к тому времени, как в Java появился собственная библиотека для логирования, все уже пользовались библиотекой log4j. История появления логирования в Java на самом деле очень долгая и познавательная, на досуге можешь почитать [этот пост на Хабре](#).

Короче говоря, своя библиотека логирования в Java есть, но ей почти никто не пользуется :)

Позже, когда появились несколько разных библиотек логирования, и все программисты начали пользоваться разными, возникла проблема совместимости.

Чтобы люди не делали одно и то же с помощью десятка разных библиотек с разными интерфейсами, был создан **абстрагирующий фреймворк slf4j** («Service Logging Facade For Java»).

Абстрагирующим он называется потому, что хотя ты и пользуешься классами slf4j и вызываешь их методы, под капотом у них работают все предыдущие фреймворки логирования: log4j, стандартный java.util.logging и другие.

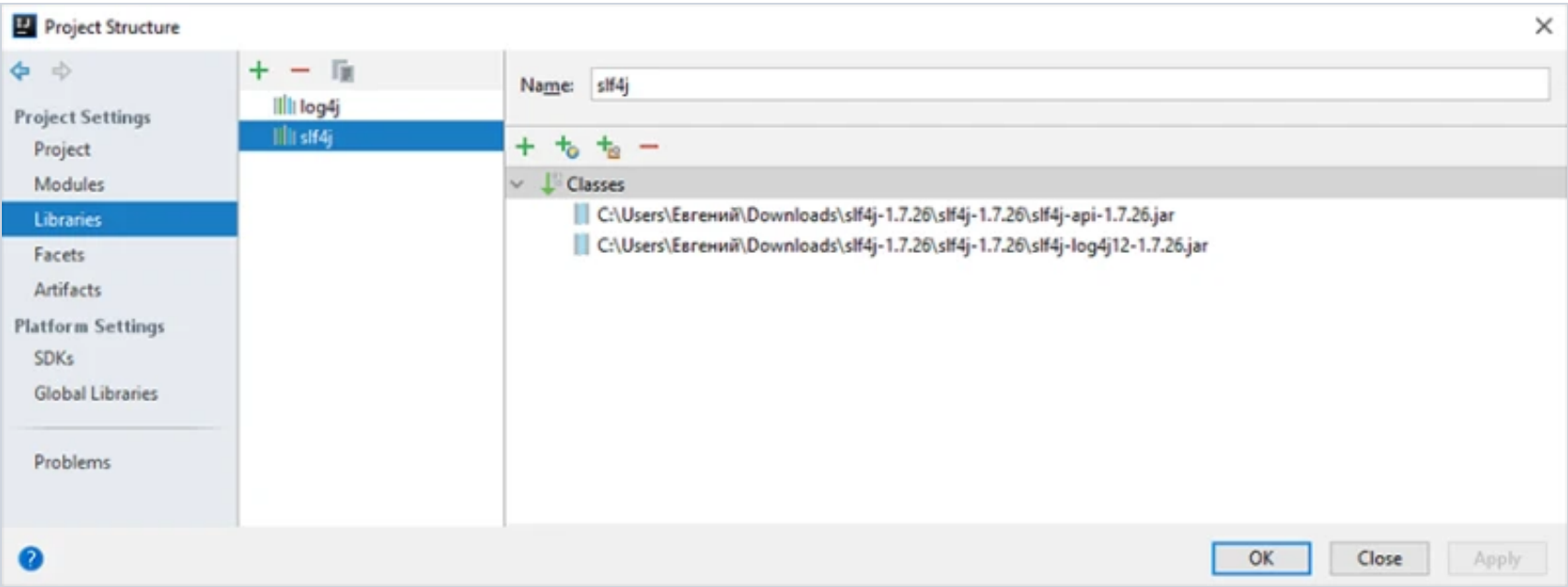
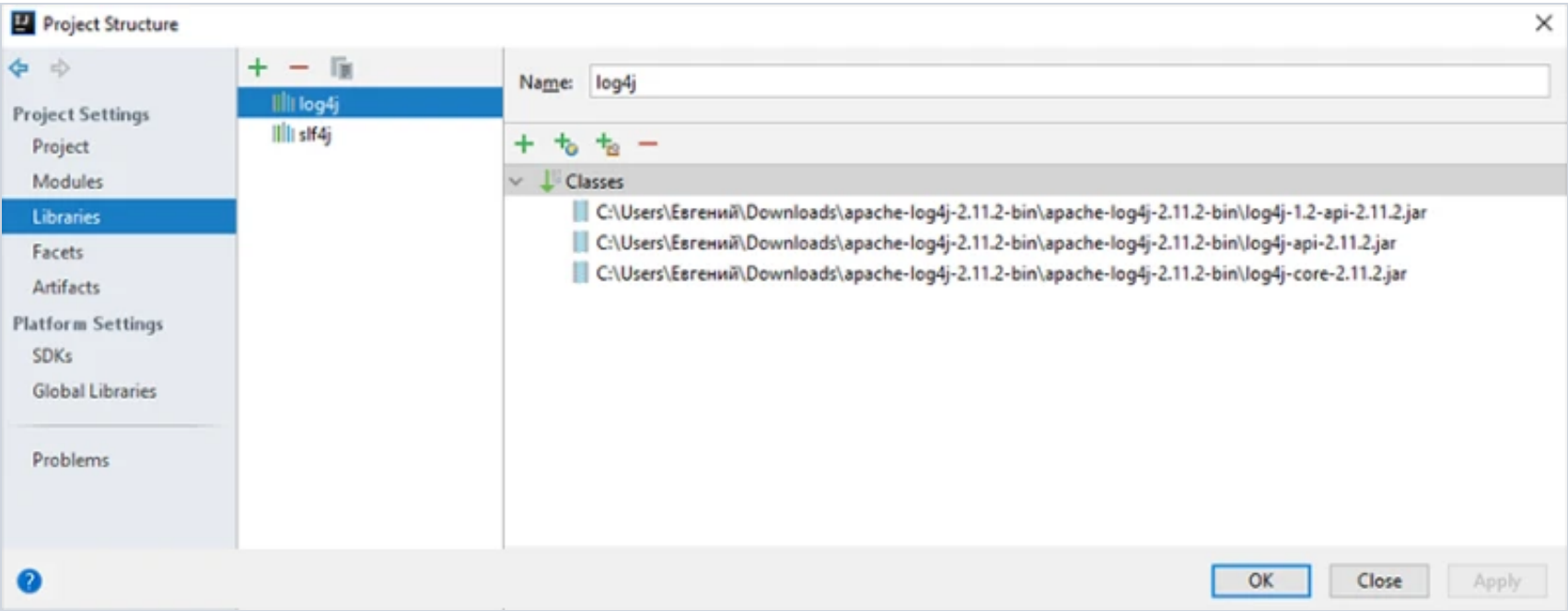
Если тебе в данный момент нужна какая-то специфическая фишка log4j, которой нет у других библиотек, но ты не хотел бы при этом жестко привязывать проект именно к этой библиотеке, просто используй slf4j. А о она уже «дернет» методы log4j.

Если ты передумашь и решишь, что фишки log4j тебе больше не нужны, тебе надо только перенастроить «обертку» (то есть slf4j) на использование другой библиотеки. Твой код не перестанет работать, ведь в нем ты вызываешь методы slf4j, а не конкретной библиотеки.

Далее архив нужно распаковать,и добавить нужные нам jar-файлы в classpath через IntelliJ IDEA.

Пункты меню: *File -> Project Structure -> Libraries*

Выбираешь нужные jar-ники и добавляешь в проект (в архивах, которые мы скачали, лежит много jar’ников, посмотри нужные на картинках)



Примечание — эта инструкция для тех студентов, которые не умеют использовать Maven. Если ты умеешь им пользоваться, лучше попробуй начать с него: это обычно намного проще

Если используешь [Maven](#), добавь такую зависимость:

1	<dependency>
2	<groupId>org.apache.logging.log4j</groupId>
3	<artifactId>log4j-slf4j-impl</artifactId>
4	<version>2.14.0</version>
5	</dependency>

Отлично, с настройками разобрались :)

Давай рассмотрим, как работает slf4j.

Как же нам сделать так, чтобы ход работы программы куда-то записывался?

Для этого нам нужны две вещи — **логгер** и **аппендер**.

Начнем с первого. **Логгер** — это объект, который полностью управляет ведением записей.

НАЧАТЬ ОБУЧЕНИЕ

Запустим наш код:

```
1  import org.slf4j.Logger;
2  import org.slf4j.LoggerFactory;
3
4  public class MyTestClass {
5
6      public static final Logger LOGGER = LoggerFactory.getLogger(MyTestClass.class);
7
8      public static void main(String[] args) {
9
10         LOGGER.info("Test log record!!!");
11         LOGGER.error("В программе возникла ошибка!");
12     }
13 }
```

Вывод в консоль:

**ERROR StatusLogger No Log4j 2 configuration file found. Using default configuration (logging only errors to the console), or user programmatically provided configurations. Set system property 'log4j2.debug' to show Log4j 2 internal initialization logging. See <https://logging.apache.org/log4j/2.x/manual/configuration.html> for instructions on how to configure Log4j 2**

**15:49:08.907 [main] ERROR MyTestClass - В программе возникла ошибка!**

Что же мы тут видим?

Сначала мы видим сообщение об ошибке. Она появилась, потому что сейчас у нас не хватает необходимых настроек. Поэтому наш логгер сейчас умеет выводить только сообщения об ошибках (ERROR) и только в консоль.

Метод `logger.info()` выполнен не был. А вот `logger.error()` сработал! В консоли появилась текущая дата, метод, где возникла ошибка (`main`), слово ERROR и наше сообщение!

**ERROR — это уровень логгирования.**

В общем, если запись в логе помечена словом ERROR, значит, в этом месте программы произошла ошибка. Если запись помечена словом INFO — значит это просто текущая информация о нормальной работе программы.

В библиотеке SLF4J довольно много разных уровней логгирования, которые позволяют гибко настроить ведение журнала.

Управлять ими очень легко: вся необходимая логика уже заложена в класс `Logger`.

Тебе достаточно просто вызывать нужные методы. Если ты хочешь залогировать обычное сообщение, вызывай метод `logger.info()`. Сообщение об ошибке — `logger.error()`. Вывести предупреждение — `logger.warn()`

Теперь поговорим об **аппендере**.

**Аппендер — это место, куда приходят твои данные.** Можно сказать, противоположность источнику данных — «точка В».

По умолчанию данные выводятся в консоль. Обрати внимание, в предыдущем примере нам не пришлось ничего настраивать: текст появился в консоли сам, но при этом логгер из библиотеки log4j умеет выводить в консоль только сообщения уровня ERROR.

Чтобы изменить поведение логгера по умолчанию, нам нужно сконфигурировать свой **файловый аппендер**.

Для начала, прямо в папке src нужно создать файл **log4j.xml**, или в папке resources, если используешь Maven, or in the resources folder, in case you use Maven.

С форматом xml ты уже знаком, у нас недавно была [лекция](#) про него :)

Вот таким будет его содержимое:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Configuration status="INFO">
3      <Appenders>
4          <File name="MyFileAppender" fileName="C:\Users\Username\Desktop\testlog.txt" immediateFlush="true">
5              <PatternLayout pattern="%d{yyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
6          </File>
7      </Appenders>
8      <Loggers>
9          <Root level="INFO">
10             <AppenderRef ref="MyFileAppender"/>
11          </Root>
12      </Loggers>
13  </Configuration>
```

Выглядит не особо-то и сложно :)

Но давай все-таки пройдемся по содержимому.

```
<Configuration status="INFO">
```

Это так называемый status-logger. Он не имеет отношения к нашему логгеру и используется во внутренних процессах log4j.

Можешь установить status="TRACE" вместо status="INFO", и в консоль будет выводиться вся информация о внутренней работе log4j (status-logger выводит данные именно в консоль, даже если наш аппендер для программы будет файловым). Нам это сейчас не нужно, поэтому оставим все как есть.

```
1  <Appenders>
2      <File name="MyFileAppender" fileName="C:\Users\Евгений\Desktop\testlog.txt" append="true">
3          <PatternLayout pattern="%d{yyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
4      </File>
5  </Appenders>
```

Тут мы создаем наш аппендер.

Тег `<File>` указывает что он будет файловым.

`name="MyFileAppender"` — имя нашего аппендера.

`fileName="C:\Users\Username\Desktop\testlog.txt"` — путь к лог-файлу, куда будут записываться все данные.

`append="true"` — нужно ли дозаписывать ли данные в конец файла. В нашем случае так и будет. Если установить значение *false*, при каждом новом запуске программы старое содержимое лога будет удаляться.

форматирования. Здесь мы с помощью регулярных выражений можем настраивать формат текста в нашем логе.

```
1  <Loggers>
2      <Root level="INFO">
3          <AppenderRef ref="MyFileAppender"/>
4      </Root>
5  </Loggers>
```

Здесь мы указываем уровень логгирования (root level).

У нас установлен уровень INFO: то есть, все сообщения уровней выше INFO (по таблице, которую мы рассматривали выше) в лог не попадут.

У нас в программе будет 3 сообщения: одно INFO, одно WARN и одно ERROR. С текущей конфигурацией все 3 сообщения будут записаны в лог. Если ты поменяешь значение root level на ERROR, в лог попадет только последнее сообщение из `LOGGER.error()`.

Кроме того, сюда же помещается ссылка на аппендер. Чтобы создать такую ссылку, нужно внутри тега `<Root>` создать тег `<AppenderRef>` и добавить ему параметр `ref="имя твоего аппендера"`.

Имя аппендера мы создали вот тут, если ты забыл:

```
<File name="MyFileAppender"
```

А вот и код нашей программы!

```
1  import org.slf4j.Logger;
2  import org.slf4j.LoggerFactory;
3
4  public class MyTestClass {
5
6      public static final Logger LOGGER = LoggerFactory.getLogger(MyTestClass.class);
7
8      public static void main(String[] args) {
9
10         LOGGER.info("Начало работы программы!!!");
11
12         try {
13             LOGGER.warn("Внимание! Программа пытается разделить одно число на другое");
14             System.out.println(12/0);
15         } catch (ArithmeticException x) {
16
17             LOGGER.error("Ошибка! Произошло деление на ноль!");
18         }
19     }
20 }
```

Он, конечно, немного кривоватый (перехват `RuntimeException` — идея так себе), но для нашей целей отлично подойдет :)

Давай запустим наш метод `main()` 4 раза подряд и посмотрим на наш файл `testlog.txt`. Создавать его заранее не нужно: библиотека сделает это автоматически.

Для запуска:

НАЧАТЬ ОБУЧЕНИЕ



Теперь у тебя есть настроенный логгер. Ты можешь поиграться с какими-то написанными тобой ранее программами, добавив вызовы логгера во все методы, и посмотреть на получившийся журнал:)

В качестве дополнительного чтения очень рекомендую тебе [вот эту статью](#).

Там тема логирования рассмотрена углубленно, и за один раз прочитать ее будет непросто. Но в ней содержится очень много полезной дополнительной информации.

Например, ты научишься конфигурировать логгер так, чтобы он создавал новый текстовый файл, если наш файл testlog.txt достиг определенного размера:)

А наше занятие на этом завершено! Ты сегодня познакомился с очень важной темой, и эти знания точно пригодятся тебе в дальнейшей работе.

До новых встреч! :)

−

+87

+

Комментарии (31)

популярные

новые

старые

JavaCoder

Введите текст комментария

Kim    Уровень 36, Амстердам, Netherlands

7 июля, 21:01    ⋮

У вас ошибка. SLF4j - это Simple Logging , у вас он Service...

Ответить

−

0

+

Макс Дудин    Уровень 39, Калининград, Россия

29 марта, 00:54    ⋮

всё таки я его подключил, не прошло и года... =) и даже частично работает

Ответить

−

0

+

Zheleznyak Maxim    Уровень 40, Moscow, Россия

3 февраля, 17:51    ⋮

Добрый день,  
  
Подскажите пожалуйста.  
  
как добавить файл конфигурации Logger`а?  
  
способ описанный в лекции (https://javarush.ru/groups/posts/2293-zachem-nuzhno-logirovanie) не работает.  
сейчас удалил файл и хочу добиться:  
  
ERROR StatusLogger No Log4j 2 configuration file found. Using default configuration (logging only errors to the console), or user programmatically provided configurations. Set system property 'log4j2.debug' to show Log4j 2 internal initialization logging. See https://logging.apache.org/log4j/2.x/manual/configuration.html for instructions on how to configure Log4j 2  
  
Как создать и явно указать файл конфигурации?  
Сейчас тупо выводит сообщения в консоль:

НАЧАТЬ ОБУЧЕНИЕ

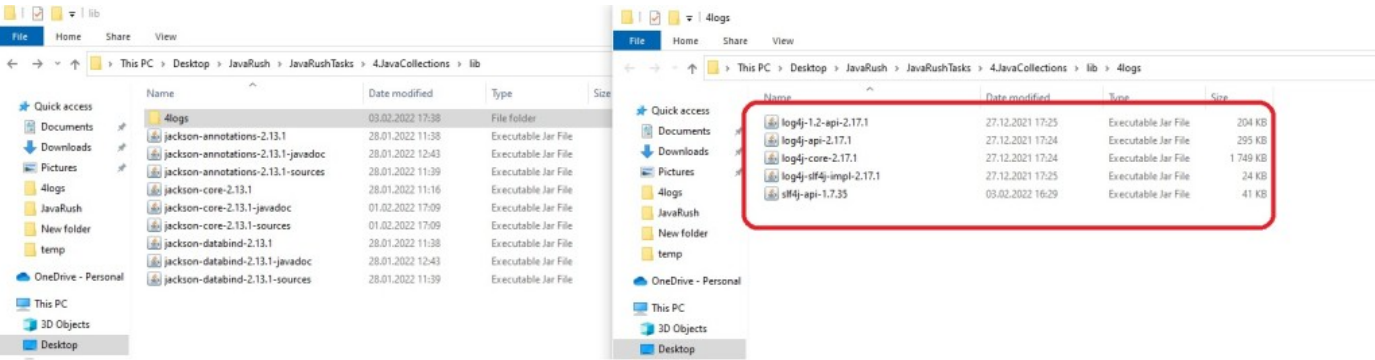
```
C:\Users\zhn\OneDrive\Desktop\corretto-1.8.0_302\bin\java.exe ...
17:40:09.697 [main] ERROR com.javarush.task.task34.task3412.MyTestClass - Внимание! Программа пытается разделить одно число на другое
17:40:09.702 [main] ERROR com.javarush.task.task34.task3412.MyTestClass - Ошибка! Произошло деление на ноль!

Process finished with exit code 0
```

Мой код:

```
1 package com.javarush.task.task34.task3412;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5
6 public class MyTestClass {
7     public static final Logger LOGGER = LoggerFactory.getLogger(MyTestClass.class)
8
9
10    public static void main(String[] args) {
11        LOGGER.info("Начало работы программы!!!");
12        LOGGER.error("Внимание! Программа пытается разделить одно число на другое"
13        LOGGER.error("Ошибка! Произошло деление на ноль!");
14
15
16    }
17 }
```

Набор подключенных библиотек:



Ответить 0

**AlinaAlina** Уровень 35, Санкт-Петербург 5 февраля, 18:34

Скачиваете и добавляете 5 jar'ов, например, как на картинках в статье  
Потом (если не Maven) в src кладёте log4j2.xml (а не log4j.xml) с тем содержанием, что в статье  
И всё ок)

Ответить 0

**Pineapple** Уровень 45, Абакан, Россия 9 августа 2021, 12:31

```
1 <File name="MyFileAppender" fileName="C:\test\testlog.txt" immediateFlush="false"
```

Ответить +2

**Михаил** Уровень 37, Россия 31 марта 2021, 06:08



Тут ошибка. Все сообщения уровней **НИЖЕ** INFO в лог не попадут.

Ответить 

− +3 +

**Edil Kalmamatov** Уровень 35 23 сентября 2021, 11:45 

...

ни выше, ни ниже, потому что тут идет отсылка к таблице, которой нет  
но подозреваю, что речь идет о таблице из статьи  
<https://javarush.ru/quests/lectures/questcollections.level04.lecture09?post=full>  
там сверху вниз идет перечисление от ALL до OFF. Тогда все правильно - сообщения уровней  
выше INFO(по таблице) в лог не попадут.

Ответить 

− 0 +

**Илья** Backend Developer в **СберТех** 8 февраля 2021, 18:00 

...

ссылка на статью по прежнему не работает

Ответить 

− 0 +

**Dmitriy** Уровень 14, Москва, Москва 3 февраля 2021, 12:51 

...

Добрый день!  
Ссылка на дополнительную статью в конце этой статьи не работает!

Ответить 

− 0 +

**Luicich** Уровень 39, Минск 20 января 2021, 13:14 

...

В качестве дополнительного чтения очень рекомендую тебе вот эту статью.  
ОТЛИЧНАЯ СТАТЬЯ!!!

Ответить 

− +3 +

**KIRA** Уровень 41 EXPERT 11 января 2021, 16:26 

...

Оставлю ссылку на нужные jar файлы для тех кто не знает где скачать, как скачать или просто лень  
искать  
[google disk](#)  
Куда жать  
Скачать можете куда угодно, лично я положил файлы в папку lib  
Нажимаем Shift + Ctrl+ Alt + S  
Выбираем Libraries, жмем + Java и выбираем место расположение jar файлов.

По примеру выше 3 log4 файла и еще раз + Java, чтобы добавить 2 slf4j

Переходим в раздел Modules  
Выбираем нужный модуль, в нашем случае Collections, меню Dependencies и проверяем наши  
библиотеки

Выбираем нужные и жмем добавить

В конце не забудьте нажать Apply  
После этого все должно корректно работать

Ответить 

− +10 +

**Илья** Backend Developer в **СберТех** 8 февраля 2021, 13:03 

...

красава! уважаю!!!

Ответить 

− 0 +

**Roman Grand** Уровень 35, Новосибирск, Россия 4 августа 2021, 19:50 

...

добавлю, xml файл с настройками должен называться **log4j2.xml**, а не ~~log4j.xml~~, так как  
используется второе поколение log4j, а не устаревшее первое, которое описывается в данной  
статье

Ответить 

− +5 +

ОБУЧЕНИЕ

Курсы программирования

Курс Java

Помощь по задачам

Подписки

Задачи-игры

СООБЩЕСТВО

**Валерьян** Уровень 41, Сыктывкар 8 ноября 2020, 01:27 

...

Подскажите, пожалуйста, если вот такая ошибка возникает, что я не докачал?

Ответить 

− 0 +

Форум

Отзывы

Чат Показать еще комментарии

FAQ

Истории успеха

Поддержка


Активности

НАЧАТЬ ОБУЧЕНИЕ

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский

▼

