



# Инструменты логирования

- **Log4j** - полноценное решение, которое создавалось из потребностей разработчиков, но так и не попало в JDK.
- **JUL — `java.util.logging`** - включен в JDK, но им никто не пользуется.
- **JCL — `jakarta commons logging`** — когда не было единого логгера, придумали JCL, как единую обертку над разными логгерами. Но ей все равно не пользовались.
- **Logback** - open-source приемник Log4j. Та же идея в основе, но улучшена производительность, расширены опции фильтрации и добавлена нативная поддержка SLF4J. Базовая функциональность работает без каких-либо настроек.
- **SLF4J — `simple logging facade for java`** — обертка от создателей Log4j, которая объединила под собой Log4j, JUL, JCL и Logback.

# Уровни логирования

Уровни логирования – условное разграничение сообщений с целью ранжирования логов.

Уровни логирования на примере log4j.

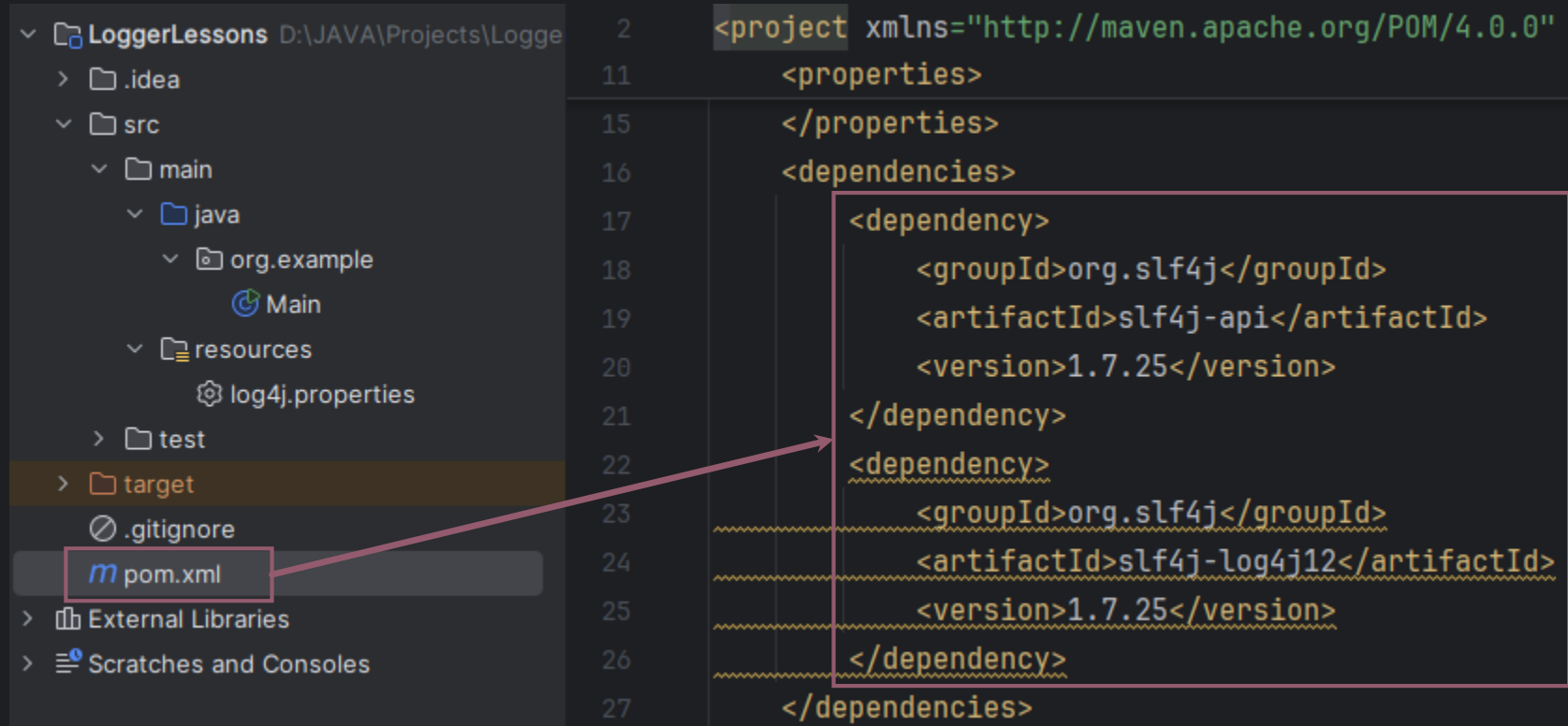
- OFF** - никакие логи не записываются, все будут проигнорированы;
- FATAL** - ошибка, после которой приложение уже не сможет работать и будет остановлено, например, JVM out of memory error;
- ERROR** - уровень ошибок, когда есть проблемы, которые нужно решить. Ошибка не останавливает работу приложения в целом. Остальные запросы могут работать корректно;
- WARN** - обозначаются логи, которые содержат предостережение. Произошло неожиданное действие, несмотря на это система устояла и выполнила запрос;
- INFO** - лог, который записывает важные действия в приложении. Это не ошибки, это не предостережение, это ожидаемые действия системы;
- DEBUG** - логи, необходимые для отладки приложения. Для уверенности в том, что система делает именно то, что от нее ожидают, или описания действия системы: “method1 начал работу”;
- TRACE** - менее приоритетные логи для отладки, с наименьшим уровнем логирования;
- ALL** - уровень, при котором будут записаны все логи из системы.

# Уровни логирования

|                     | Тип лога |       |       |      |      |       |       |
|---------------------|----------|-------|-------|------|------|-------|-------|
| Уровень логирования |          | FATAL | ERROR | WARN | INFO | DEBUG | TRACE |
|                     | OFF      |       |       |      |      |       |       |
|                     | FATAL    | x     |       |      |      |       |       |
|                     | ERROR    | x     | x     |      |      |       |       |
|                     | WARN     | x     | x     | x    |      |       |       |
|                     | INFO     | x     | x     | x    | x    |       |       |
|                     | DEBUG    | x     | x     | x    | x    | x     |       |
|                     | TRACE    | x     | x     | x    | x    | x     | x     |
|                     | ALL      | x     | x     | x    | x    | x     | x     |

# Использование связки SLF4J+Log4j

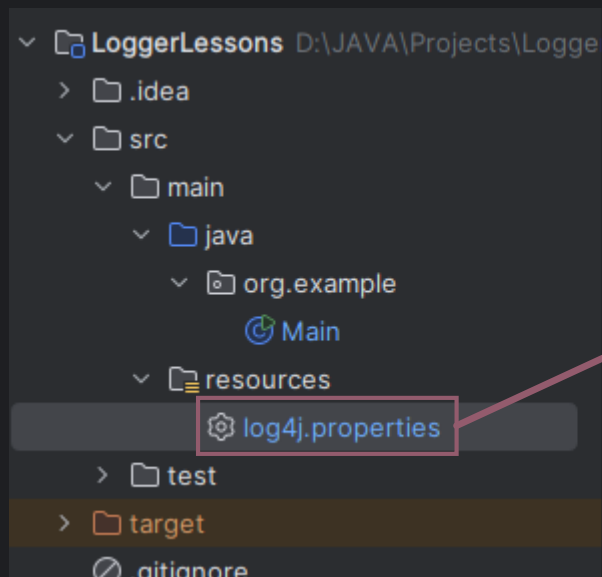
## Шаг 1. Настройка зависимостей



```
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
11 <properties>
15 </properties>
16 <dependencies>
17 <dependency>
18 <groupId>org.slf4j</groupId>
19 <artifactId>slf4j-api</artifactId>
20 <version>1.7.25</version>
21 </dependency>
22 <dependency>
23 <groupId>org.slf4j</groupId>
24 <artifactId>slf4j-log4j12</artifactId>
25 <version>1.7.25</version>
26 </dependency>
27 </dependencies>
```

# Использование связки SLF4J+Log4j

## Шаг 2. Настройка Log4j



```
1 #Root logger options
2 log4j.rootLogger=ALL, CONSOLE
3
4 #CONSOLE appender settings
5 #Appender class
6 log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
7 #Log level
8 log4j.appender.CONSOLE.threshold=DEBUG
9
10 #Layout pattern of message
11 log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
12 log4j.appender.CONSOLE.layout.ConversionPattern=%d [%-5p] : %c:%L : %m%n
```

**Узел логирования** — сущность, соответствующая пакету приложения, поддерживает точечную настройку логирования и формирует иерархию узлов.

**RootLogger** — корневой узел логирования

**Аппендер** — инструмент для записи логов.

**ConsoleAppender** - для получения данных в консоль приложения

**RollingFileAppender** - для записи логов в файл

**JDBCAppender** - для записи логов в базу данных и т.д.

# Использование связки SLF4J+Log4j

## Шаг 3. Использование логера

```
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6
7 public class Main {  Nik *
8     public static void main(String[] args) {  Nik *
9         Logger logger = LoggerFactory.getLogger(Main.class);
10
11         logger.error("Error message");
12         logger.info("Info message");
13         logger.debug("Debug message");
14         logger.trace("Trace message");
15     }
16 }
```

```
2025-04-07 16:19:54,672 [ERROR] : org.example.Main:12 : Error message
2025-04-07 16:19:54,674 [INFO ] : org.example.Main:13 : Info message
2025-04-07 16:19:54,674 [DEBUG] : org.example.Main:14 : Debug message
```