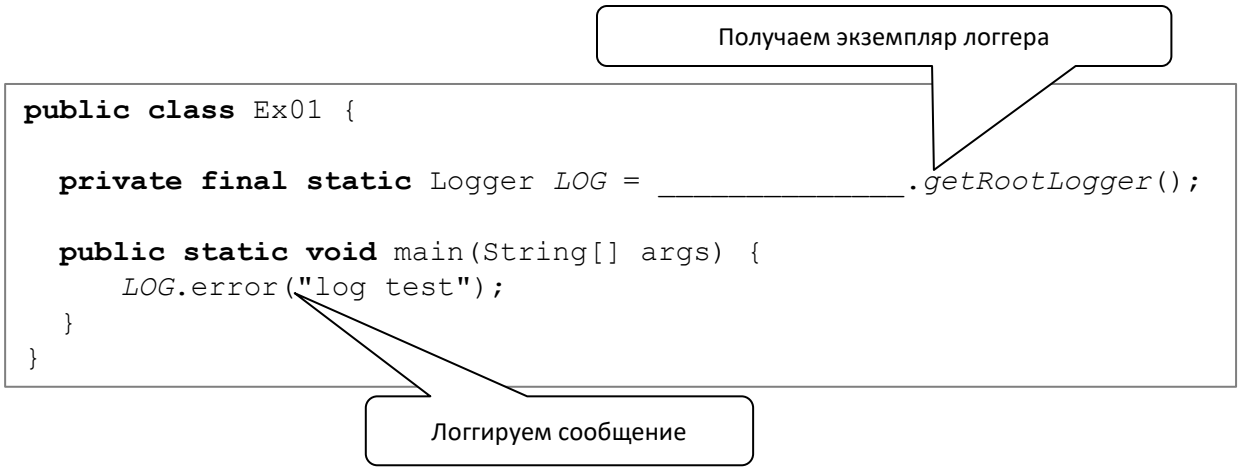
Основы логгирования. Log4j2

Принципы работы с log4j



_____
_____
_____

log4j2.xml

```
<_____ monitorInterval="3">
  <_____>
    <_____ name="STDOUT" target="SYSTEM_OUT">
      <_____ pattern="%d{ABSOLUTE} [%-5p] (%F:%L) - %m%n" />
    </Console>
  </appenders>
  <_____>
    <_____ level="warn">
      <appender-ref ref="STDOUT" />
    </root>
  </loggers>
</configuration>
```

_____
_____
_____
_____
_____
_____

API Log4j – это инструмент для формирования журнала сообщений (отладочных, информационных, сообщений об ошибках).

Логгеры - это экземпляры класса org.apache.logging.log4j.Logger, вызывая методы которых в подсистему **log4j** посылается сообщение для логгинга.

Посылаемые сообщения различаются по приоритету:

TRACE < **DEBUG** < **INFO** < **WARN** < **ERROR** < **FATAL**

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Каждому LoggerConfig-у назначается т.н. уровень логгирования (Log Level). Для вывода сообщения необходимо, чтобы _____
_____, чем уровень LoggerConfig-а.

```
<configuration>
  <Properties>
    <Property name="filename">logs/log.log</Property>
    <Property name="basePatterLayout">%d{ISO8601} [%-5p] (%F:%L) -
%m%n</Property>
  </Properties>
  <appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="${basePatterLayout}" />
    </Console>
  </appenders>
  <loggers>
    <logger name="ex.Ex02" _____ >
      <appender-ref ref="STDOUT"/>
    </logger>
    <root _____>
      <appender-ref ref="STDOUT"/>
    </root>
  </loggers>
</configuration>
```

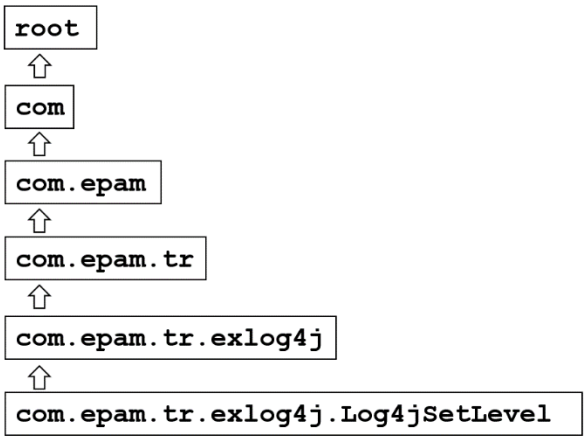
```
package ex;
public class Ex02 {

  private final static Logger LOG = LogManager.getLogger(Ex02.class);

  public static void main(String[] args) {
    LOG._____("log test 2", new Exception("Test Exception"));
    LOG.debug("debug level message");
  }
}
```

имя логгера

LoggerConfig образуют иерархию, определяемую их именами (подобно пакетам в Java).

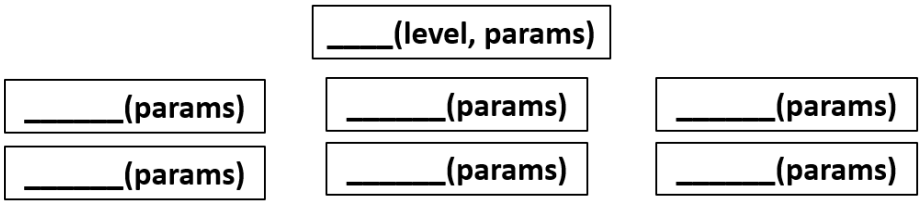


Если уровень конфигуратора не указывается, то он _____ уровень от его родителя.

Уровень корневого конфигуратора по умолчанию равен _____.

Корневой регистратор всегда существует и у него нет имени. Он может быть получен методом _____

Методы для вывода сообщений.




```
...
}catch(Exception ex){
    log.error("произошла ошибка", ex);
}
...
```

```
if (log.isDebugEnabled()) {
      log.debug("Result: "+ result);
}
```

```
log.debug("Result: {}", result);
```

_____
_____
_____
_____
_____
_____
_____
_____

Logger **наследует** все аппендеры своих родителей, если обратное не установлено свойством _____=false.

```
<configuration>
  <appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m %n" />
    </Console>
  </appenders>
  <loggers>
    <logger name="ex.Ex03" _____="false" >
      <appender-ref ref="STDOUT"/>
    </logger>
    <root level="debug">
      <appender-ref ref="STDOUT"/>
    </root>
  </loggers>
</configuration>
```

Log4j2 позволяет использовать диагностические контексты, основанные на Map, и на Stack

```
package ex;
public class Ex04 {
  private final static Logger logger =
                LogManager.getLogger(Ex04.class.getName());

  public static void main(String[] args) {
    String idUser = UUID.randomUUID().toString();
    _____.put("id", idUser);

    logger.debug("Message 1");
    logger.debug("Message 2");

    _____.clearMap();
    logger.debug("Message 3");
  }
}
```

```
<configuration>
  <appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m - id=%_______%n" />
    </Console>
  </appenders>
  <loggers>
    <logger name="ex.Ex04" level="debug" >
      <appender-ref ref="STDOUT"/>
    </logger>
    <root level="debug">
    </root>
  </loggers>
</configuration>
```

```
public class Ex04 {
  private final static Logger logger =
LogManager.getLogger(Ex04.class.getName());

  public static void main(String[] args) {
    String idUser = UUID.randomUUID().toString();
    ThreadContext._______(idUser);
    ThreadContext._______("Ivanov");
    logger.debug("Message 1");
    logger.debug("Message 2");
    ThreadContext._______();
    logger.debug("Message 3");
  }
}
```

```
<configuration>
  <appenders>
    <Console name="STDOUT" target="SYSTEM_OUT">
      <PatternLayout pattern="%m - NDC = %___%n" />
    </Console>
  </appenders>
  <loggers>
    <logger name="threadcontent.Ex04" level="debug" >
      <appender-ref ref="STDOUT"/>
    </logger>
    <root level="debug">
    </root>
  </loggers>
</configuration>
```
