Logging

myLog.log

Зачем?

1. **Отладка и поиск ошибок.** Логи помогают находить и устранять ошибки, показывая, где именно что-то пошло не так.
2. **Анализ поведения приложения.** Логи фиксируют ключевые события, такие как обработка запросов, выполнение операций с базой данных, взаимодействие с внешними сервисами.
3. **Поддержка и мониторинг.** Логирование позволяет контролировать, как ведет себя приложение на реальном сервере, и быстро реагировать на сбои.

Новичок

```java
public class MyClass {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
        } catch (Exception e) {
            System.err.println("Ошибка: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Ошибка: / by zero
```

резюме: просто вывод пойманного исключения красненьким

Уже смешарик

```java
import java.util.logging.Logger;

public class MyClass {
    2 usages
    private static final Logger logger = Logger.getLogger(MyClass.class.getName());

    public static void main(String[] args) {
        logger.info( msg: "Программа запущена");
        try {
            int result = 10 / 0;
        } catch (Exception e) {
            logger.severe( msg: "Ошибка: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
окт. 09, 2024 7:41:54 PM
MyClass main

INFO: Программа запущена

окт. 09, 2024 7:41:54 PM
MyClass main

SEVERE: Ошибка: / by zero
```

а что за Logger?

библиотеки:

- ➔ java.util.logging (встроенная в JDK)
- ➔ Log4j (от Apache Foundation)
- ➔ SLF4J (Simple Logging Facade for Java)
- ➔ Logback (модернизированная версия Log4j)

важно! Logger объявляется и инициализируется так:

```
private static final Logger logger = Logger.getLogger(ClassName.class.getName());
```

Уровни логирования

java.util.logging	
FINEST	ваще подробное сообщение жесть
FINER	подробное сообщение
FINE	сообщение об успешной операции
CONFIG	сообщение о конфигурации
INFO	информационное сообщение
WARNING	предупреждение
SEVERE	критическая ошибка

Log4j
TRACE
DEBUG
INFO
WARN
ERROR
FATAL

Как выглядят сообщения:

```java
import java.util.logging.Logger;

public class MyClass {
    3 usages
    private static final Logger logger = Logg

    public static void main(String[] args) {
        logger.severe( msg: "Критическая ошибка!");
        logger.warning( msg: "Предупреждение!");
        logger.info( msg: "Информация о просто работе просто программы");
    }
}
```

OUTPUT:

```
окт. 09, 2024 8:38:07 PM MyClass main
SEVERE: Критическая ошибка!
окт. 09, 2024 8:38:07 PM MyClass main
WARNING: Предупреждение!
окт. 09, 2024 8:38:07 PM MyClass main
INFO: Информация о просто работе просто программы
```

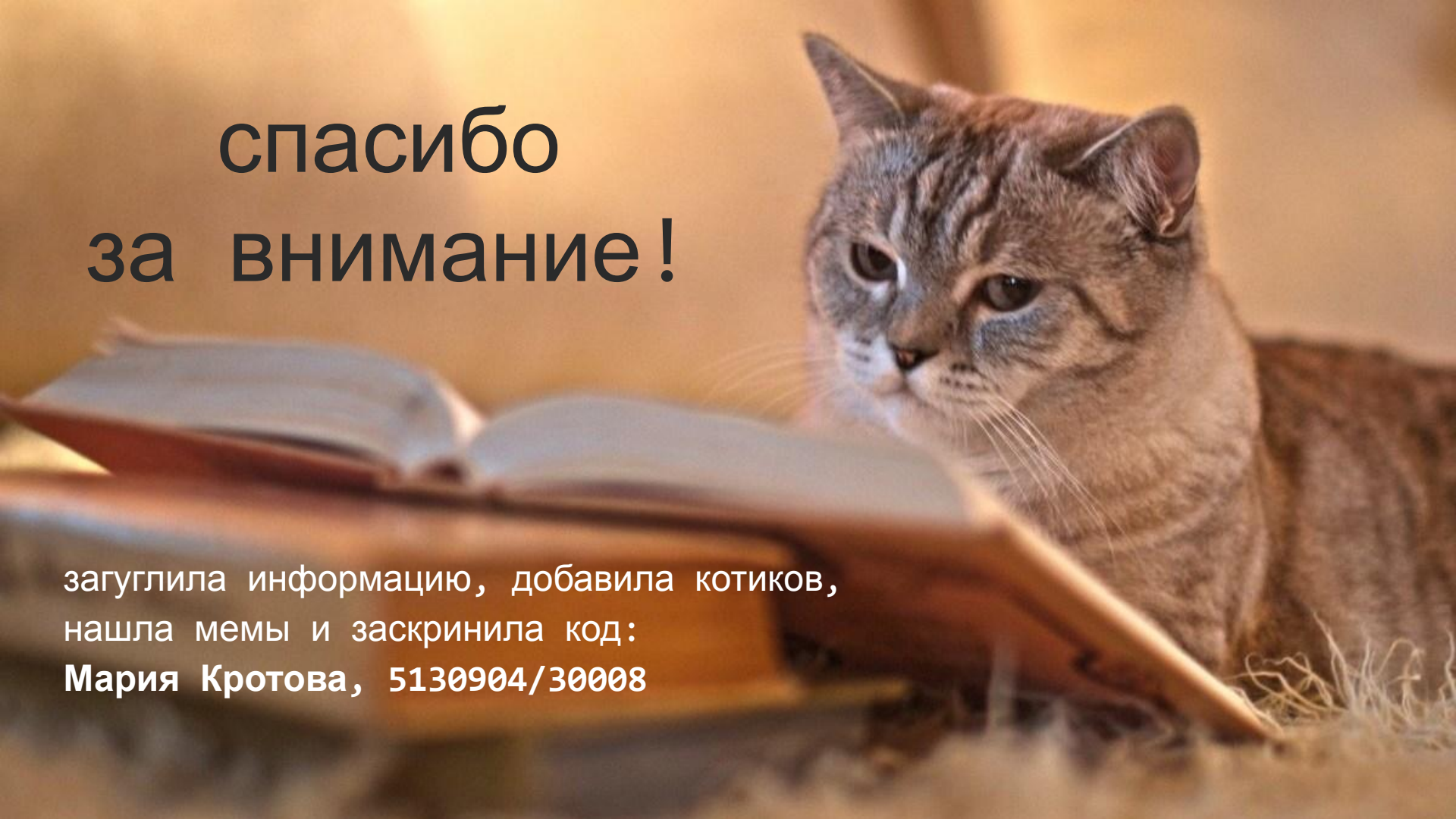
а куда?

- **Консоль:**
  как мы уже посмотрели.
  Подходит для быстрой отладки небольшого куска кода.
  НЕ подходит для большого проекта.

- **Файлы логов:**
  пример пути log-файла для Log4j2:
  `<File name="MyFile" fileName="/var/logs/myapp.log">`
  Подходит для большого проекта.
  Можно конфигурировать.

Важные мелочи

- **Логи замедляют программу**
  Решение: **ассинхронное логирование Logback** - log-msg отправляются отдельным потоком.
  Минусы решения: если плохо настроить - потеряешь логи.
- **Логи могут преполниться**
  Решение: **ротация логов** - создание самой новой версии И удаление самой старой
- **Логи могут плохо работать в многопоточных приложениях**
  Решение: использовать Logback и Log4j :)


спасибо
за внимание!
загуглила информацию, добавила котиков,
нашла мемы и заскринила код:
Мария Кротова, 5130904/30008