The power of Logging & Exception handling

AXON ACTIVE
www.axon.vn

Alibaba

AGENDA

Logging

- What's logging ?
- Why do we need logging?
- Logging Framework
- Logging Level
- Logging& performance
- Logging tips

Exception

- What is exception ?
- Why do we have to care about exceptions?
- Best practices

What is logging ?

Logging is the process of writing log messages during the execution of a program to a central place.

This logging allows you to report and persist error and warning messages as well as info messages (e.g., runtime statistics) so that the messages can later be retrieved and analyzed

Why do we need logging in project ?

Dear Server,
How’s it going? You there?

Why do we need logging in project ?

Yep, I’m here. What do you need?

Why do we need logging in project ?

I need homepage.html,

Why do we need logging in project ?

OK, wait one sec

Why do we need logging in project ?

But, nothing happend after 30s

Why do we need logging in project ?

Hello Server, are your there ?

Why do we need logging in project ?

Hello …..

Why do we need logging in project ?

What the hell ? WTF ?

When web browser and server can not working together
It’s time for Human

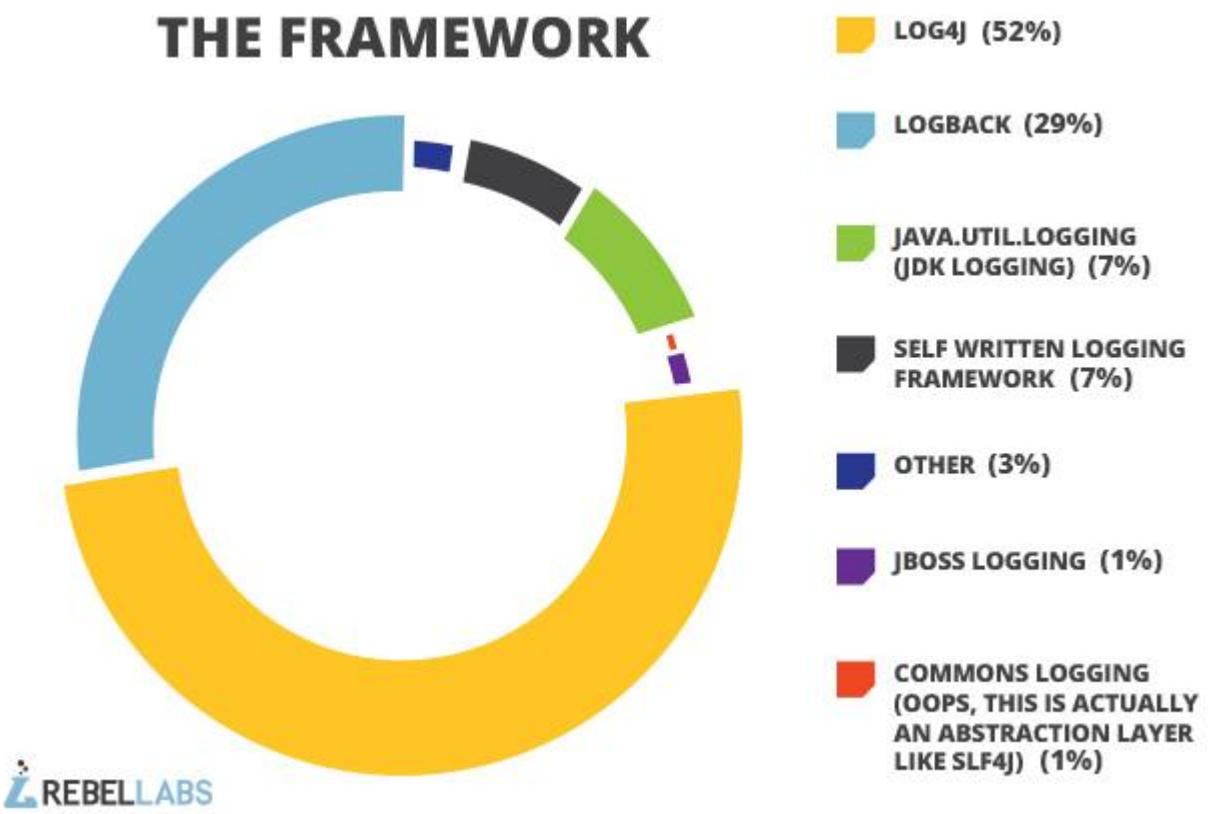
Why do we need logging in project ?

WTF, Hey developer, your website is stupid ….

Why do we need logging in project ?

Well , what should I do now ?

Logging Framework



Logging level

	LEVEL	USED TO	ENVIRONMENT
DEBUG	Lowest restricted	Write everything to **debug** an application	Development, Testing
INFO	More restricted than DEBUG	Write informative purpose, like : server has been started, start/ end function function ...	Development, Testing, Production
WARN	More restricted than INFO	Log warning sort of msg, like : connection lost between client and server, DB connection lost, Socket limit ..	Development, Testing, Production
ERROR	More restricted than WARN	Log errors and exception	Development, Testing, Production
FATAL	More restricted than ERROR	Log very serious errors which make the application will be die.	Development, Testing, Production
OFF	Highest posible rank	Turn off logging	Development, Testing, Production

https://docs.spring.io/spring-boot/docs/2.1.6.RELEASE/reference/html/boot-features-logging.html

Logging level


```java
public CompanyKey createCompany(Company companyInput) throws PermissionException,
    SystemException, ValidationException
{
    log.info("Start createCompany.");
    CompanyKey companyKey = null;

    // Validation input data
    if (companyInput == null)
    {
        log.error("The input value can not be null.");
        throw new ValidationException("The input value can not be null.");
    }

    // Read config file
    DataBaseConfigInformation dBConfigInformation = readDBConfigurationInformation();
    if (dBConfigInformation == null)
    {
        log.fatal("Can not read DB configuration data.");
        throw new ConfigurationError("Can not read DB configuration data.");
    }

    // Connect to data base
    Connection connection = openConnecttionToDB(dBConfigInformation);
    if (connection == null)
    {
        log.warn("Can not open connection to DB");
        throw new SystemException("The input value can not be null.");
    }

    // TODO (tmdung, 28 Nov 2014): create new company and return companyKey
    // ......................
    // ......................
    log.debug("Finish create company " + companyInput.getName() + " with company key is :" + companyKey);

    log.info("End createCompany.");
    return companyKey;
}
```

How logging in java effects performance ?

1- Never use DEBUG level logging in production

```
if(logger.isDebugEnabled()){
logger.debug("java logging level is DEBUG Enabled");
}
```

log.properties

```
1  ##############################################################################
2  # Log configuration
3  ##############################################################################
4
5  logger.dispatch.file.appendFile                    = false
6  logger.dispatch.file.filter                        = ALL - DEBUG
7  logger.dispatch.file.filter.ch                     = ALL - DEBUG
8  logger.dispatch.file.filter.ch.parser              = ALL - INFO - DEBUG
9  logger.dispatch.file.filter.com                    = ALL - DEBUG
10 logger.dispatch.file.filter.org                    = ALL - DEBUG
11 logger.dispatch.file.filter.java                   = ALL - INFO - DEBUG
12
```

How logging in java effects performance ?

2 - Carefully choose which kind of message & level for logging

- If you log too much information your performance will be affected
- if you don't log important information like incoming messages and outgoing messages in java logs then it would become extremely difficult to identify what happened in case of any issue or error

Tips on logging in Java

Never log important information in the log

Try to make log more helpful

Best practices

What is Exception?

Real life definition

- Someone or something that is not included in a rule, group, or list or that does not behave in the expected way. - http://dictionary.cambridge.org

Java programming definition

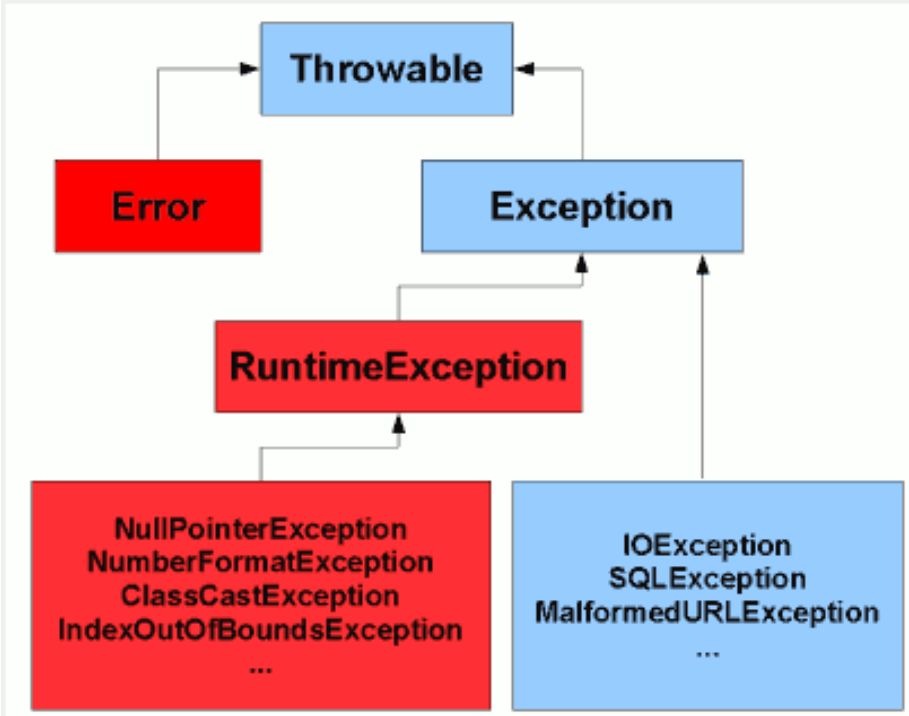
- An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. - http://docs.oracle.com

What is Exception?

- When an *exception occurs* -> *exception object*.
- Creating an exception object and handing it to the runtime system is called *throwing an exception*.

What is Exception?

- Class hierarchy.



What is Exception?

- Classical examples:
  - OutOfMemoryError (extends Error);
  - StackOverflowError (extends Error);
  - NullPointerException (extends RuntimeException);
  - IllegalArgumentException (extends RuntimeException);
  - FileNotFoundException (extends Exception);
  - IOException (extends Exception);
  - …

What is Exception?

- **Separating Error-Handling Code from "Regular" Code:**
  - Exceptions enable you to write the main flow of your code and to deal with the exceptional cases elsewhere.
- **Propagating Errors Up the Call Stack:**
  - Hence, only the methods that care about errors have to worry about detecting errors.
- **Grouping and Differentiating Error Types:**
  - Because all exceptions thrown within a program are objects, the grouping or categorizing of exceptions is a natural outcome of the class hierarchy.

Why we have to care about exceptions?

- Sometimes we have the exceptions right in our business.
- For example, here is the requirement from an PO:

Why we have to care about exceptions?

Hey gui, why I couldn't buy weapons from your site?

Because you are an exception in our business

Why we have to care about exceptions?

AXON ACTIVE
www.axon.vn

```
/**
 * Process a request from a customer.
 * Throws {@link NotSupportedCountryException} in case the customer is Russian.
 *
 * @param customer
 * @param boughtInfo
 * @throws NotSupportedCountryException
 */
public void processRequest(Customer customer, Map<Weapon, Integer> boughtInfo)
    throws NotSupportedCountryException
{
    // Don't sell for Russian.
    if (isRussian(customer))
    {
        throw new NotSupportedCountryException();
    }

    // Sells the weapons to that customer
    sellWeapons(customer, boughtInfo);
}
```

Why we have to care about exceptions?

- Also from the APIs.

```
/**
 * Private Constructor since this class is singleton.
 */
public void writeTempFile()
{
    FileInputStream in=new FileInputStream(new File(FILE_PATH));
}
```

Unhandled exception type FileNotFoundException

2 quick fixes available:

Add throws declaration

Surround with try/catch

Press 'F2' for focus

Why we have to care about exceptions?

- Almost cases are from programmer mistakes.

type Exception report

message java.lang.NullPointerException

description The server encountered an internal error that prevented it from fulfilling this request.

exception

```
org.jboss.resteasy.spi.UnhandledException: java.lang.NullPointerException
        org.jboss.resteasy.core.ExceptionHandler.handleApplicationException(ExceptionHandler.java:76)
        org.jboss.resteasy.core.ExceptionHandler.handleException(ExceptionHandler.java:212)
        org.jboss.resteasy.core.SynchronousDispatcher.writeException(SynchronousDispatcher.java:149)
        org.jboss.resteasy.core.SynchronousDispatcher.invoke(SynchronousDispatcher.java:372)
        org.jboss.resteasy.core.SynchronousDispatcher.invoke(SynchronousDispatcher.java:179)
        org.jboss.resteasy.plugins.server.servlet.ServletContainerDispatcher.service(ServletContainerDispatcher.java:220)
        org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher.service(HttpServletDispatcher.java:56)
        org.jboss.resteasy.plugins.server.servlet.HttpServletDispatcher.service(HttpServletDispatcher.java:51)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```

root cause

Why we have to care about exceptions?

- They might come from the simplest function!

```
/**
 * Check if two int number are equal.
 *
 * @param a
 * @param b
 * @return true if a equals b, otherwise, return false.
 */
public boolean isEqual(int a, int b)
{
    return a == b;
}
```

Why we have to care about exceptions?

- Exceptions are not free... so they are expensive :-) - http://stackoverflow.com/questions/567579/how-expensive-are-exceptions

Why do we have to care about exceptions?

- Consider the following example:

```
/**
 * Trims a input {@link String}.
 *
 * @param rawString
 * @return a trimmed string.
 */
public String tryTrim(String rawString)
{
    try
    {
        return rawString.trim();
    }
    catch (Exception e)
    {
        // Ignore it
    }

    return null;
}
```

- This is an expensive way to trim a String.

Why do we have to care about exceptions?

- Here is a cheaper way:

```
/**
 * Trims a input {@link String}.
 *
 * @param raw
 * @return  a trimmed string.
 */
public String ifTrim(String raw)
{
    if (raw == null)
    {
        return null;
    }

    return raw.trim();
}
```

Why do we have to care about exceptions?

✓ Programmer error → for example, NullPointerException!!!!

Best practices

1. **Never use exceptions for ordinary control flow**

DON'T DO THIS

```
/**
 * Trims a input {@link String}.
 *
 * @param rawString
 * @return a trimmed string.
 */
public String tryTrim(String rawString)
{
    try
    {
        return rawString.trim();
    }
    catch (Exception e)
    {
        // Ignore it
    }

    return null;
}
```

Best practices

1. **Never use exceptions for ordinary control flow**

DO IT THIS WAY

```
/**
 * Trims a input {@link String}.
 *
 * @param raw
 * @return  a trimmed string.
 */
public String ifTrim(String raw)
{
    if (raw == null)
    {
        return null;
    }

    return raw.trim();
}
```

Best practices

2. **Declare the specific checked exceptions that your method can throw**

DON'T DO THIS

```
public void foo() throws Exception { //Incorrect way
}
```

This is much better

```
public void foo() throws SpecificException1, SpecificException2 { //Correct way
}
```

Best practices

3. **Do not catch the Exception class rather catch specific sub classes, never catch Throwable class**

DON'T DO THIS

```
try {
    someMethod();
} catch (Exception e) {
    LOGGER.error("method has failed", e);
}
```

Best practices

4. **Don’t let the stack trace lost.**

```
catch (NoSuchMethodException e) {
   throw new MyServiceException("Some information: " + e.getMessage());  //Incorrect way
}
```

```
catch (NoSuchMethodException e) {
   throw new MyServiceException("Some information: " , e);  //Correct way
}
```

Best practices

5. **Always catch only those exceptions that you can actually handle**

```
catch (NoSuchMethodException e) {
   throw e; //Avoid this as it doesn't help anything
}
```

Best practices

6. **Throw early, catch late**

- Failing fast by throwing exceptions as soon as an error is detected can eliminate the need to construct objects or open resources
- The trick is to catch exceptions at the proper layer, where your program can either meaningfully recover from the exception and continue without causing further errors, or provide the user with specific information, including instructions on how to recover from the error.


TIME FOR QUESTIONS

thank you!