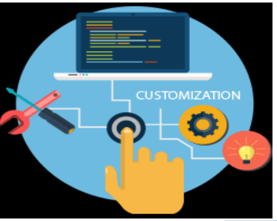
Error And Exception Handling In Quality Context

NIC Webinar- Knowledge Sharing among peers PAN INDIA
on
7th FEB 2019

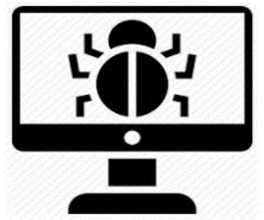
Pooja Singh
Software Quality Group
Scientist-D, NIC HQ

Why Exception and Error Handling Crucial?



Large scale software is too complex to be bug free, even with the most thorough testing process, we test only specific situations.

Bugs lead to both errors and exceptions. It's important to understand the differences between errors and exceptions in application.



A proactive approach can be taken to maintain a healthy application for both developers and end users.

What if Errors are not handled properly?

If the failure is ignored and processing continues assuming everything is good, this **can lead to corrupted data, and other hard to reproduce exceptions.**

The developer team will **not be able to cope up with Runtime problems.**

Code will not be clean, **understanding and fixing issues** will **consume developer's time**.

Recovering from errors and exceptions as well as logging exceptions is very important to target **Reliable** and **Robust** software

What is an Error ?

An error is a term used to describe any issue that arises unexpectedly and results in an incorrect output.

Types of Errors

Syntactic Error	**Occur due to poor understanding of the language.**
Logical Error	**Occur due to poor understanding of problem or solution procedure.**

Best Practices to Handle Errors

- ✓ Errors can usually be avoided with simple checks

- ✓ If simple checks won’t suffice, design the application such that the situation can be handled gracefully.

What is an exception?

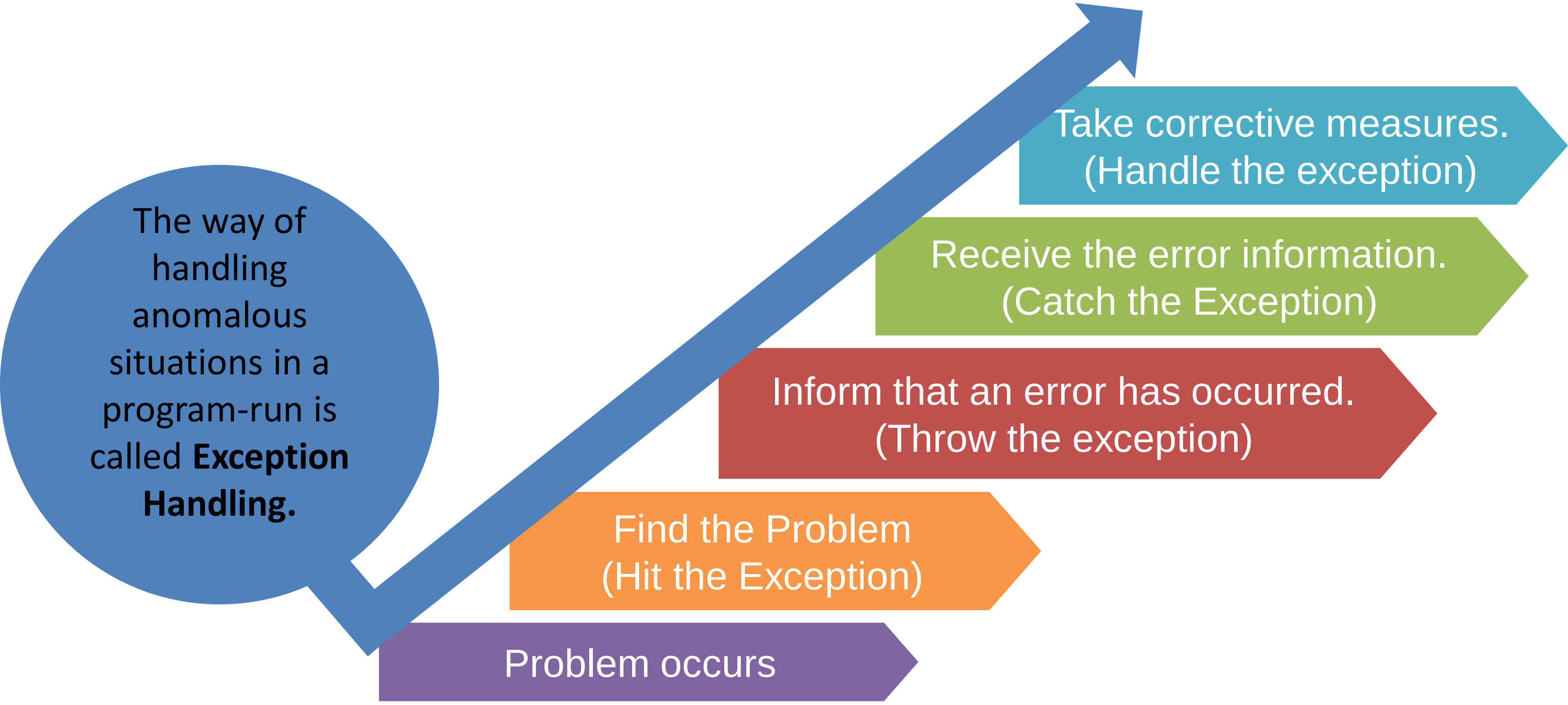
Exceptions are **run time anomalies** or **unusual conditions** that a program may encounter while executing.

Some common examples like division by zero, opening a non-existent file, hard disk failure.

Types of Exceptions

Synchronous exceptions	Asynchronous exceptions
The exceptions which occur during the program execution due to some fault in the input data are known as synchronous exceptions.	The exceptions caused by events or faults unrelated (external) to the program and beyond the control of the program are called asynchronous exceptions.
Example: errors such as out of range, overflow	Example: errors such as keyboard interrupts, hardware malfunctions, disk failure.

Exception Handling


The way of handling anomalous situations in a program-run is called **Exception Handling.**
Take corrective measures. (Handle the exception)
Receive the error information. (Catch the Exception)
Inform that an error has occurred. (Throw the exception)
Find the Problem (Hit the Exception)
Problem occurs

Writing Robust Web Applications

When writing a function, **throw exceptions for the management** instead of returning a boolean.

Remember **“Throw Early Catch Late”** principle.

- Exception more likely to be **thrown in low level methods.**

- Catch late means Catch exceptions at the most convenient place.

- e.g. If an invalid parameter is passed then the layer that provided the parameter should deal with the consequences.

Writing Robust Web Applications

Wrapping an exception? Do not lose the previous exception!

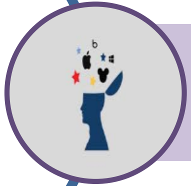
Pass on the root exception.

Validate user input to **catch adverse conditions very early** in request processing

Writing Robust Web Applications

Pass all relevant information to exceptions to make them informative as much as possible.

Use finally blocks instead of catch blocks if you are not going to handle exception.



In case of Custom Exceptions are thrown, the names must be clear and meaningful.

Exception Handling Anti-Patterns

Never use Exceptions for flow control

- Do not manage business logic with exceptions. It makes code hard to read, understand and ugly. Use conditional statements instead. .

Either log the exception or throw it but never do both

- Logging and throwing will result in multiple log messages in log files

Exception Handling Anti-Patterns

Never throw any exception from finally block

Do not Handle Exceptions inside Loops

Exception Handling Anti-Patterns

Never Eat the Exception in Catch Block

Do not overuse Exception Handling. Minimize catching unnecessary exceptions

Error and Exception Logging

For large scale software may use logging tools for improving the tracing and handling of unexpected situations.

- Asp.net: log4Net
- Java: Log4j2, LogBack, TinyLog
- PHP: ApacheLog4PHP

To Conclude…..

"It is an error to not handle an exception."

Henning Thielemann

Exceptions are costly in terms of performance, if not handled smartly.

Thank you !

support-sqg@nic.in

IP No: 5294