Kotlin Try Catch

Kotlin Try Catch Statement

Kotlin Try Catch is used to handle the code that could throw an exception at run-time. Enclose the block of code, that could throw an exception, in try block, and handle the exceptions using catch clauses that follow try block.

If an exception thrown at run-time is not handled, it could terminate the program. So, learning how to handle exceptions is a must learn concept.

Following is the syntax of try-catch block in Kotlin.

```
try {
   //code that code potentially throw and exception
 } catch(e: Some_Exception) {
   //code to handle if this exception is occurred
 } catch(e: Exception) {
   //you can have multiple catch blocks
   //code to handle if this exception is occurred
 }
```

So, for a try-catch block, you can have one try block and one or more catch blocks. Also, in each catch block, you can specify a specific exception as argument, to catch that type of exception.

Example 1 – Try Catch in Kotlin

In this example, we will try a classic divide by zero exception. We shall enclose the code that throws divide by zero exception in try block, and the code to handle this exception in catch block.

Kotlin Program – example.kt

```
fun main(args: Array<String>) {
    val a = 6
    val b = 0
    var c : Int

    try {
        c = a/b
    } catch (e : Exception){
        println("Exception is handled.")
    }
```

```
}
```

Output

```
Exception is handled.
```

In the above program, the catch block has argument of type Exception class. So, literally any type of Exception could be caught by this catch block and the code inside this block is executed.

After catching the exception, the execution continues with the subsequent statement after try-catch, if any.

Kotlin – Catch Specific Exception

Instead of generalizing, you can specify a specific Exception like ArithmeticException, IOException, etc, as argument for the catch block.

In the below example, instead of Exception class type, we specify that the catch block has to handle ArithmeticException class type object.

Kotlin Program – example.kt

```
fun main(args: Array<String>) {
    val a = 6
    val b = 0
    var c : Int

    try {
        c = a/b
    } catch (e : ArithmeticException){
        println("a is $a and b is $b. a/b ?? You cannot divide something with zero.")
    }
}
```

Output

```
a is 6 and b is 0. a/b ?? You cannot divide something with zero.
```

Also, please note that if an Exception of type other than `ArithmeticException` is thrown by the code in try block, in this case, an error is thrown in the run-time and the execution is halted.

Kotlin Exception – Stack Trace

Stack Trace helps you track down the piece of code that caused Exception.

Stack trace could be accessed from exception object thrown to catch block. To print the stack trace, use

Stack trace could be accessed from exception object thrown to catch block. To print the stack trace, use printStackTrace() method on the exception object.

Kotlin Program – example.kt

```
fun main(args: Array<String>) {
    val a = 6
    val b = 0
    var c : Int

    try {
        c = a/b
    } catch (e : Exception){
        e.printStackTrace()
    }
}
```

Output

```
java.lang.ArithmeticException: / by zero
    at exception_handling.ChangeValKt.main(ChangeVal.kt:8)
```

Notice that in the stack trace, the exception thrown is ArithmeticException and caused by the code at line 8. When you are not catching the Specific Exception, catch block that handles Exception class object gets executed.

Kotlin Exception – Message

Message of the Exception caused could be accessed using **message** property of the exception object.

Kotlin Program – example.kt

```
fun main(args: Array<String>) {
    val a = 6
    val b = 0
    var c : Int

    try {
        c = a/b
    } catch (e : ArithmeticException){
        println(e.message)
    }
}
```

Output

```
/ by zero
```

Multiple Catch Blocks

Multiple Catch Blocks

Based on the type of exceptions thrown by the code in try block, there could be multiple catch blocks following try block.

In the following Kotlin program, there is a try-catch block with multiple blocks.

Kotlin Program – example.kt

```
import java.io.IOException

fun main(args: Array<String>) {
    val a = 6
    val b = 0
    var c : Int

    try {
        c = a/b
        print(c)
    } catch (e : IOException){
        println("An IOException occurred. Please Handle.")
    } catch (e : Exception){
        println("An Exception occurred. Handle.")
    }
}
```

Run this program and observe the output.

Output

```
An Exception occurred. Handle.

Process finished with exit code 0
```

The exception thrown by the try block is ArithmeticException. In the catch section, there are two catch blocks. The first one handles IOException, which is not a specific match. So, the control goes to the second catch block. This catch block handles any Exception. So, the second catch block gets executed.

Conclusion

In this Kotlin Tutorial – Kotlin Try Catch, we have learned how to write a try catch block, how exceptions are thrown by the code in try block to the respective catch block, how to access information like stack trace, message, etc., from exception object.

- Kotlin Tutorial

Getting Started

- Setup Kotlin(Java) Project
- Kotlin Example Program
- Convert Java to Kotlin
- Kotlin Main Function
- Kotlin Loops
- Kotlin For Loop
- Kotlin While, Do While Loops
- Kotlin Repeat
- Kotlin Ranges
- Kotlin When

Object Oriented Concepts

Classes

- Kotlin - Class, Primary and Secondary Constructors
- Kotlin Sealed Class
- Kotlin Data Class
- Kotlin Enum
- Kotlin - Extension Functions

Inheritance

- Kotlin Inheritance
- Kotlin Override Method of Super Class

Abstraction

- Kotlin Abstraction
- Kotlin Abstract Class
- Kotlin - Interfaces
- Kotlin Null Safety

Exception Handling

- **Kotlin Try Catch**
- Kotlin Throw Exception

- Kotlin Custom Exception

Fix Compilation Errors

- Kotlin - Variable must be initialized
- Kotlin - Primary Constructor call expected
- Kotlin - Null can not be a value of a non-null type String
- Kotlin - Cannot create an instance of an abstract class

Kotlin - String Operations

- Kotlin - Compare Strings
- Kotlin - Replace String
- Kotlin - Split String
- Kotlin - Split String to Lines
- Kotlin - String Capitalize

Kotlin - Functions

- Kotlin Function - Default Arguments
- Kotlin - Use Function

Kotlin Collections

Kotlin List

- Kotlin List
- Kotlin List forEach

Kotlin File Operations

- Kotlin - Create File
- Kotlin - Read File
- Kotlin - Read File as List of Lines
- Kotlin - Write to File
- Kotlin - Append Text to File
- Kotlin - Check if File Exists
- Kotlin - Copy a File to Other
- Kotlin - Iterate through all files in a directory

- Kotlin - Delete Recursively
- Kotlin - Get File Extension

Kotlin Interview Q/A

- Kotlin Interview Questions

Kotlin Android

- Kotlin Android Tutorial

Useful Resources

- How to Learn Programming