Dart Try Catch

Dart Try Catch

In Dart programming language, you can handle exceptions or errors using try catch statement.

Dart code can throw objects of type Exception, Error or any non-null object.

In this tutorial, we will learn how to use `try-catch` to handle exceptions, `on` keyword to catch specific Errors or Exceptions and what `finally` keyword does with try-catch statement.

Try Catch

Try Catch statement allows you to handle Errors or Exceptions without actually exiting the execution when an Exception or Error occurs.

All you have to do is, surround the piece of code, you expect to throw any Error or Exception with try block. Then if at all an exception occurs, the object is sent to the catch block. You can access the exception object and continue with the program execution. In catch block, you can write program statements.

In the following example, we have taken a list initialized with three elements. In the try block, we are trying to print out the elements of list within index range of `[0,10)`. When we are trying to access fourth element, which is not present in the list, an Error is thrown. catch block catches this thrown object. And you can access this object in the catch block.

Dart Program

```
void main(){
    var myList = [52, 6, 87];

    try {
        for(var i=0;i<10;i++) {
            print(myList[i]);
        }
    } catch (e) {
        print('Something happened while printing the list');
        print('Printing out the message: $e');
    }
    print('Continuing with the rest of the program..');
}
```

Output

```
52
6
87
Something happened while printing the list
Printing out the message: RangeError (index): Invalid value: Not in range 0..2, inclusiv
Continuing with the rest of the program..
```

After the catch block, the program execution continues.

Try – on – catch

catch block can handle any time of Exception or Error. But if you want to handle a specific type of Error or Exception in some different way, you can use on keyword.

Each **on**-block can handle specified Exception or Error.

In the following example, we have an **on**-block to handle RangeError. If at all the code in try-block throws a RangeError, the on block with RangeError is executed and then the control comes out of try-catch block. After try-catch, execution continues with rest of the program.

Dart Program

```
void main(){
    var myList = [52, 6, 87];

    try {
        for(var i=0;i<10;i++) {
            print(myList[i]);
        }
    } on RangeError {
        print('You are coming out of range for the list. Check the range you are conside
    } catch (e) {
        print('Something unknown exception happened while printing the list');
    }
    print('Continuing with the rest of the program..');
}
```

Output

```
52
6
87
You are coming out of range for the list. Check the range you are considering
Continuing with the rest of the program..
```

finally

If you would like to execute a set of statements for sure, whatever happened with the try-on-catch blocks, then finally is the keyword. Like try, on or catch blocks, you can write a finally block, but only at the end of try catch block.

Dart Program

```
void main(){
    var myList = [52, 6, 87];

    try {
        for(var i=0;i<10;i++) {
            print(myList[i]);
        }
    } on RangeError {
        print('You are coming out of range for the list. Check the range you are conside
    } catch (e) {
        print('Something unknown exception happened while printing the list');
    } finally {
        print('This finally block will execute.');
    }
    print('Continuing with the rest of the program..');
}
```

Output

```
52
6
87
You are coming out of range for the list. Check the range you are considering
This finally block will execute.
Continuing with the rest of the program..
```

Conclusion

In this Dart Tutorial, we learned about try, on, catch and finally keywords and how to use these to handle Errors or Exceptions in Dart.

Dart Programming

- Dart Tutorial
- Install Dart on Windows
- Dart - Hello World
- Dart - Variables

- Dart - Comments
- Dart - If Else
- Dart - For Loop

Dart String Operations

- Dart - Concatenate Strings
- Dart - Split String
- Dart - Replace Substring in String
- Dart - Find Substring of String
- Dart - String Length
- Dart - Trim String

Dart Exception Handling

- **Dart - Try Catch**

Dart List Operations

- Dart - List
- Dart List - Iterate
- Dart - Check if List is Empty
- Dart - Check if List Contains Element
- Dart Reverse List
- Dart Join Lists
- Dart - Check Equality of Two Lists