Learning to program with F#

Jon Sporring

August 5, 2016

Contents

Chapter 11

Exceptions

Exceptions are runtime errors, which may be handled gracefully by F#. Exceptions are handled by the `try` keyword both in expressions and computation expressions,

```
expr = ...
  | "try" expr "with" rules
  | "try" expr "finally" expr
  | ...
comp-expr = ...
  | "try" comp-expr "with" comp-rules
  | "try" comp-expr "finally" expr
  | ...
```

As an example is integer division by zero,

```
let div enum denom =
   try
      enum / denom
   with
      | :? System.DivideByZeroException -> System.Int32.MaxValue

printfn "3 / 1 = %d" (div 3 1)
printfn "3 / 0 = %d" (div 3 0)
```

```
3 / 1 = 3
3 / 0 = 2147483647
```

Listing 11.1: exceptionDivByZero.fsx - A division by zero is caught and a default value is returned.

```
> let div enum denom =
-   try
-     Some (enum / denom)
-   with
-     | :? System.DivideByZeroException -> None;;

val div : enum:int -> denom:int -> int option

>
- let a = div 3 1;;

val a : int option = Some 3

> let b = div 3 0;;
```

Attribute	Description
`System.ArithmeticException`	Failed arithmetic operation.
`System.ArrayTypeMismatchException`	Failed attempt to store an element in an array failed because of type mismatch.
`System.DivideByZeroException`	Failed due to division by zero.
`System.IndexOutOfRangeException`	Failed to access an element in an array because the index is less than zero or equal or greater than the length of the array.
`System.InvalidCastException`	Failed to explicitely convert a base type or interface to a derived type at run time.
`System.NullReferenceException`	Failed use of a `null` reference was used, since it required the referenced object.
`System.OutOfMemoryException`	Failed to use `new` to allocate memory.
`System.OverflowException`	Failed arithmetic operation in a checked context which caused an overflow.
`System.StackOverflowException`	Failed use of the internal stack caused by too many pending method calls, e.g., from deep or unbounded recursion.
`System.TypeInitializationException`	Failed initialization of code for a type, which was not caught.

Table 11.1: Built-in exceptions.

```
val b : int option = None
```

Listing 11.2: fsharpi, Option types can be used, when the value in case of exceptions is unclear.

Exceptions are a basic-type called `exn`, and F# has a number of built-in, see Table 11.1, and the user may construct new exceptions using the syntax,

```
exception-defn = [attributes] "exception" union-type-case-data [attributes]
    exception ident "=" long-ident
```

Exceptions are raised with the keywords `failwith`, `invalidArg`, `raise`, and `reraise`

```
exception DontLikeFive of string

let div enum denom =
  if denom = 5 then
    raise (DontLikeFive "5 sucks")
  try
    Some (enum / denom)
  with
    | :? System.DivideByZeroException -> None

printfn "3 / 1 = %A" (div 3 1)
printfn "3 / 0 = %A" (div 3 0)
printfn "3 / 5 = %A" (div 3 5)
```

```
3 / 1 = Some 3
3 / 0 = <null>
FSI_0001+DontLikeFive: Exception of type 'FSI_0001+DontLikeFive' was thrown.
  at FSI_0001.div (Int32 enum, Int32 denom) <0x7063e60 + 0x0015f> in <filename
       unknown>:0
  at <StartupCode$FSI_0001>.$FSI_0001.main@ () <0x70632e0 + 0x001f7> in <
      filename unknown>:0
```

```
  at (wrapper managed-to-native) System.Reflection.MonoMethod:InternalInvoke (
      System.Reflection.MonoMethod,object,object[],System.Exception&)
  at System.Reflection.MonoMethod.Invoke (System.Object obj, BindingFlags
      invokeAttr, System.Reflection.Binder binder, System.Object[] parameters,
       System.Globalization.CultureInfo culture) <0x227c270 + 0x000a1> in <
      filename unknown>:0
Stopped due to error
```

Listing 11.3: exceptionDefinition.fsx - A user-defined exception is raised but not caught by outer construct.

Remember

- exn type Spec-4.0 Chapter 18.1
- Spec-4.0 Section 18.2.8

...

Bibliography

[1] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366, 1932.

[2] Programming Research Group. Specifications for the ibm mathematical formula translating system, fortran. Technical report, Applied Science Division, International Business Machines Corporation, 1954.

[3] John McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.

[4] X3: ASA Sectional Committee on Computers and Information Processing. American standard code for information interchange. Technical Report ASA X3.4-1963, American Standards Association (ASA), 1963. `http://worldpowersystems.com/projects/codes/X3.4-1963/`.

[5] George Pólya. *How to solve it*. Princeton University Press, 1945.

Index