

道语言异常处理

September 19, 2009

道异常类被组织为树形结构，其基类是 Exception:

```
class Exception
{
  routine Exception( content='' ){ Content = content }
  protected
    var Rout = '';
    var File = '';
    var FromLine = 0;
    var ToLine = 0;
  public
    var Name = 'Exception';
    var Content : any = 'undefined exception';
}
```

目前支持的类包括:

类	异常类型
Exception	Any
Exception.None	None
Exception.Any	Any or none
Exception.Error	Any error
Exception.Warning	Any warning
Exception.Error.Field	invalid field accessing
Exception.Error.Field.NotExist	field not exist
Exception.Error.Field.NotPermit	field not permit
Exception.Error.Float	invalid floating point operation
Exception.Error.Float.DivByZero	division by zero
Exception.Error.Float.Overflow	floating point overflow
Exception.Error.Float.UnderFlow	floating point underflow
Exception.Error.Index	invalid index
Exception.Error.Index.OutOfRange	index out of range
Exception.Error.Key	invalid key
Exception.Error.Key.NotExist	key not exist
Exception.Error.Param	invalid parameter list for the call
Exception.Error.Syntax	invalid syntax
Exception.Error.Type	invalid variable type for the operation
Exception.Error.Value	invalid variable value for the operation

从以上类可以派生出新的异常类。

任何运行时错误将抛出异常类实例。程序也可在任何地方使用raise语句抛出异常类实例。被抛出的异常可由rescue语句捕获，然后进行处理，相关语法是：

```
stmt_tryrescue ::=
  'try' '{' stmt_block
  { '}' 'rescue' [ '(' expr ')' ] '{' stmt_block }*
```

这里expr可以为空，或是一个或多个异常类。如果rescue后没有expr，所有异常都将被捕获。每个try-rescue块里，可以有多个rescue语句，每个rescue语句都将被检测是否可以捕获到异常，如果捕获到了，相应的语句块将被执行，并且被捕获的异常将可用全局变量exceptions列表访问。

如果所有rescue语句都被检测了，但还剩没有捕获的异常，那么当前运行函数

将中止，并将异常抛给起调用者。如果当前函数没有调用者，那么程序将被中止，并打印异常信息。在一个`rescue`语句块里，在修改某些运行条件后，`retry`可用来重新尝试执行`try`语句块里的代码。

例子，

```
try{
  raise Exception.Error( "error test" ), Exception.Warning( "warning test" ),
  Exception( "exception" );
}rescue( Exception.Error, Exception.Warning ){
  stdio.println( exceptions );
}rescue{
  stdio.println( "rescued" );
}
```