FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

ITSE333A ABAP

Lesson 7: Exception Handling

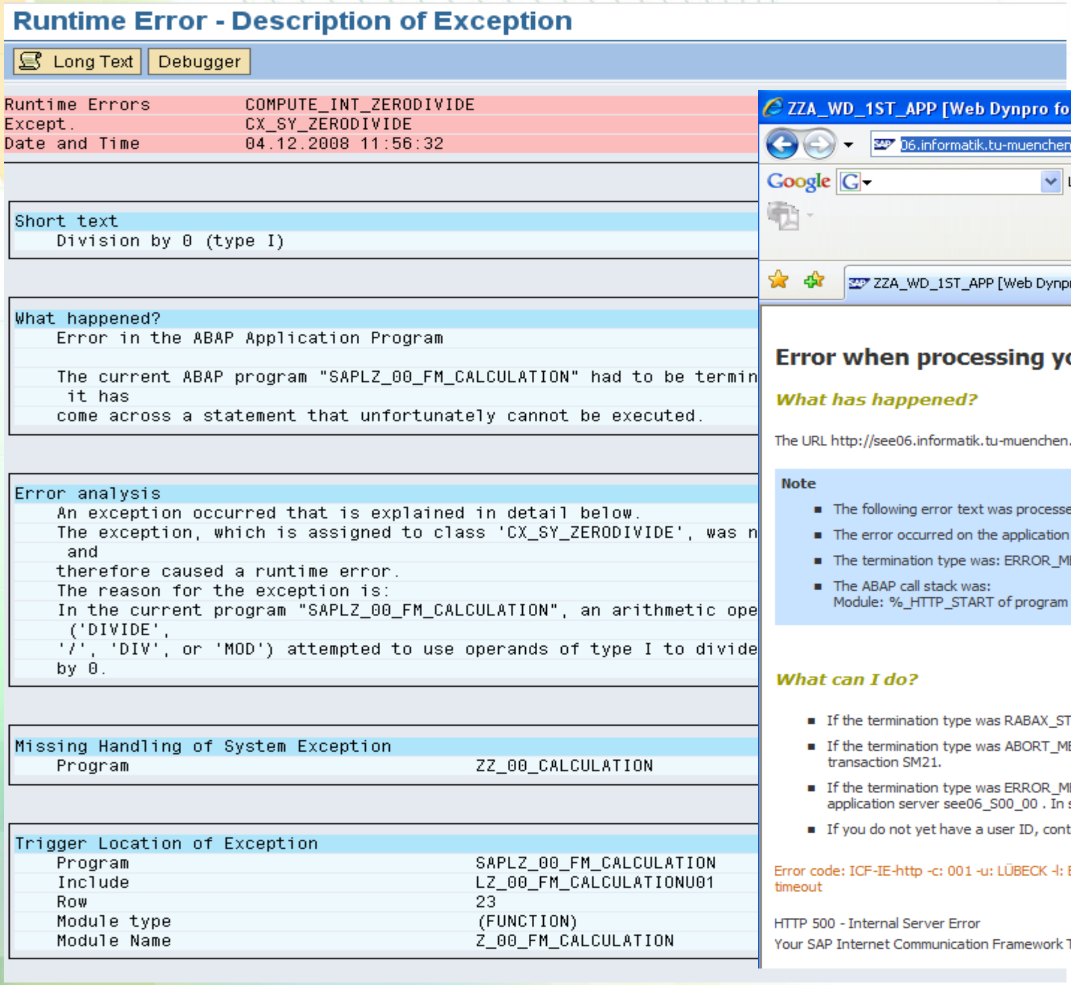
Anthony D. Aquino

Agenda

1. Motivation
2. Principles of exception handling
3. Raising exceptions
4. Catching exceptions
5. Exception classes
6. Exception subclasses
7. Declaration of exceptions
8. Categories of exceptions
9. Exception texts
10. Example

Motivation – Sample exceptions

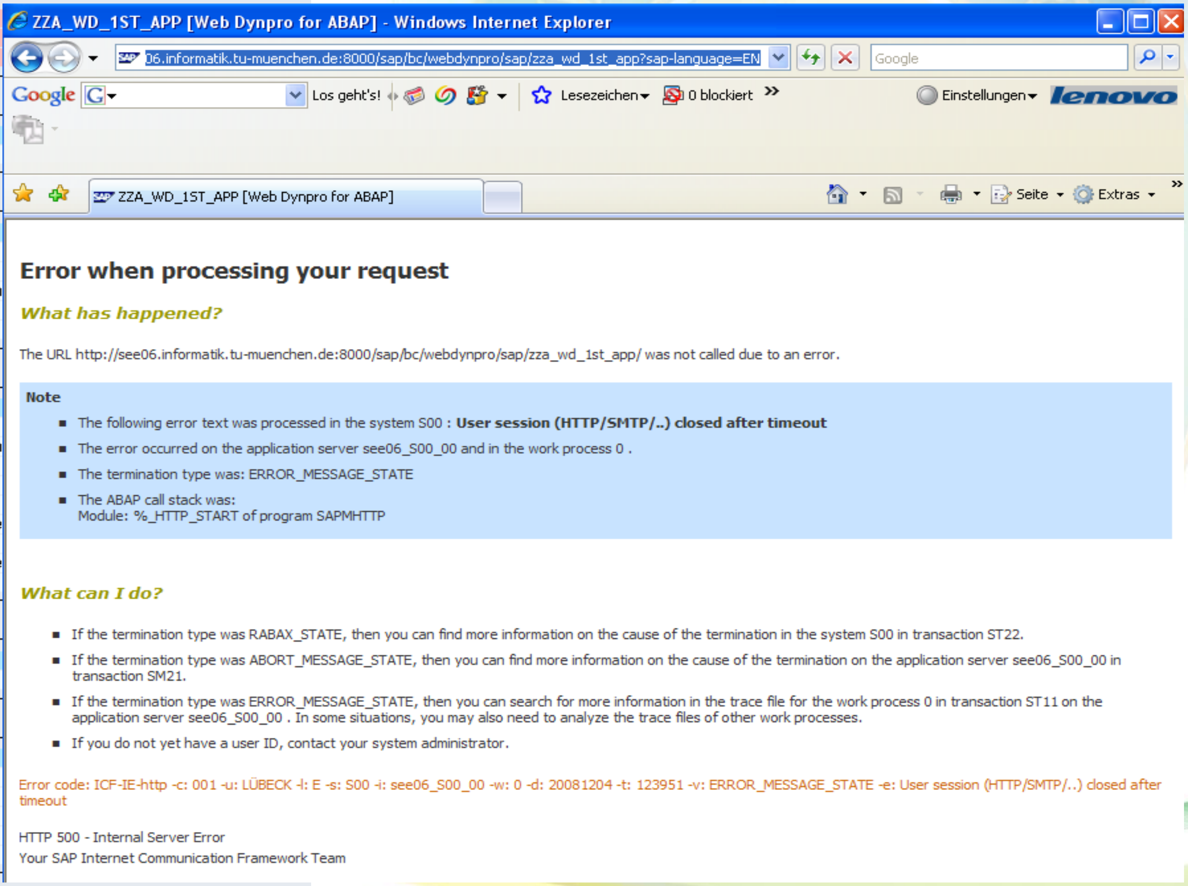
Runtime error – ABAP program / SAP Gui



Runtime Error - Description of Exception

```
Runtime Errors          COMPUTE_INT_ZERODIVIDE
Except.                 CX_SY_ZERODIVIDE
Date and Time           04.12.2008 11:56:32

Short text
    Division by 0 (type I)

What happened?
    Error in the ABAP Application Program

    The current ABAP program "SAPLZ_00_FM_CALCULATION" had to be termin
     it has
    come across a statement that unfortunately cannot be executed.

Error analysis
    An exception occurred that is explained in detail below.
    The exception, which is assigned to class 'CX_SY_ZERODIVIDE', was n
     and
    therefore caused a runtime error.
    The reason for the exception is:
    In the current program "SAPLZ_00_FM_CALCULATION", an arithmetic ope
     ('DIVIDE',
    '/', 'DIV', or 'MOD') attempted to use operands of type I to divide
    by 0.

Missing Handling of System Exception
    Program                                 ZZ_00_CALCULATION

Trigger Location of Exception
    Program                                 SAPLZ_00_FM_CALCULATION
    Include                                 LZ_00_FM_CALCULATIONU01
    Row                                     23
    Module type                             (FUNCTION)
    Module Name                             Z_00_FM_CALCULATION
```

Runtime error – BSP



ZZA_WD_1ST_APP [Web Dynpro for ABAP] - Windows Internet Explorer
06.informatik.tu-muenchen.de:8000/sap/bc/webdynpro/sap/zza_wd_1st_app?sap-language=EN
Google
Los geht's!
Lesezeichen
0 blockiert
Einstellungen
lenovo
ZZA_WD_1ST_APP [Web Dynpro for ABAP]
Seite
Extras

Error when processing your request

What has happened?

The URL http://see06.informatik.tu-muenchen.de:8000/sap/bc/webdynpro/sap/zza_wd_1st_app/ was not called due to an error.

Note

- The following error text was processed in the system S00 : **User session (HTTP/SMTP/..) closed after timeout**
- The error occurred on the application server see06_S00_00 and in the work process 0 .
- The termination type was: ERROR_MESSAGE_STATE
- The ABAP call stack was:
  Module: %_HTTP_START of program SAPMHTTP

What can I do?

- If the termination type was RABAX_STATE, then you can find more information on the cause of the termination in the system S00 in transaction ST22.
- If the termination type was ABORT_MESSAGE_STATE, then you can find more information on the cause of the termination on the application server see06_S00_00 in transaction SM21.
- If the termination type was ERROR_MESSAGE_STATE, then you can search for more information in the trace file for the work process 0 in transaction ST11 on the application server see06_S00_00 . In some situations, you may also need to analyze the trace files of other work processes.
- If you do not yet have a user ID, contact your system administrator.

Error code: ICF-IE-http -c: 001 -u: LÜBECK -l: E -s: S00 -i: see06_S00_00 -w: 0 -d: 20081204 -t: 123951 -v: ERROR_MESSAGE_STATE -e: User session (HTTP/SMTP/..) closed after timeout

HTTP 500 - Internal Server Error
Your SAP Internet Communication Framework Team

Motivation

- If an error occurs, that the ABAP runtime cannot resolve, an exception is thrown.
- Without exception handling, the complete stack is rolled back and the error is presented to the user.
- With exception handling you can
  - present the error to the user in a more user-friendly way,
  - react on the error,
  - cleanup (free any used resources before terminating),
  - or even continue executing the program ignoring the error.

Principles of exception handling

- Exceptions are used to handle unexpected events during execution.
- An exception can be handled locally or by any calling service in the stack.
- This is useful since the calling service can react on an exception and keep the program running in a consistent state.

Raising exceptions

- Exceptions can be raised with implicit object creation

```
RAISE EXCEPTION TYPE cx_flight_not_found
EXPORTING flightid = ‘LH221’.
```

- …or with explicit object creation

```
DATA exception TYPE REF TO cx_flight_not_found.
CREATE OBJECT exception
EXPORTING flightid = ‘LH221’.
RAISE EXCEPTION exception.
```

- …or raised by kernel (aka runtime exception)

`x = 1 / 0.` (this creates the exception cx_sy_zerodivide)

Catching exceptions

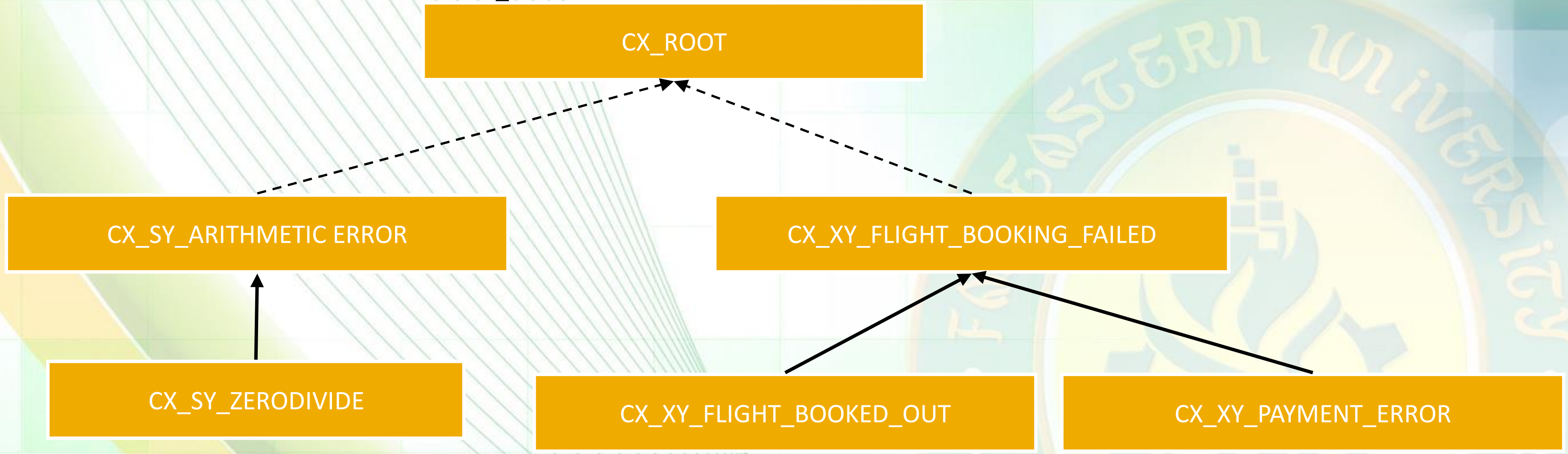
```
TRY.
“- Protected Area
“- any statements here
CATCH cx_a1 [INTO exception1 ].
“- handler code for exception cx_a1
CATCH cx_a2 [INTO exception2 ].
“- handler code for exception cx_a2
ENDTRY.
```

- Any statement between TRY and first CATCH is in the protected area.
- If a statement in the protected area raises an exception the ABAP runtime checks if an appropriate handler is present.
- A handler (CATCH-Block) handles all exceptions for the given exception class and all subclasses.
- If no local handler is found, the exception will be passed up the stack to the calling service.
- If no handler is found in the stack, the application dumps. (ABAP Runtime will display a short dump and terminate)

Exception classes

- Each exception class can define its own attributes
- Methods inherited from cx_root:
  - get_source_position:
    - Returns position where exception has been raised
  - get_text, get_longtext:
    - Returns textual description of exception
- Global exception classes have an automatically generated constructor containing one optional parameter for each non-private attribute

Exception subclasses



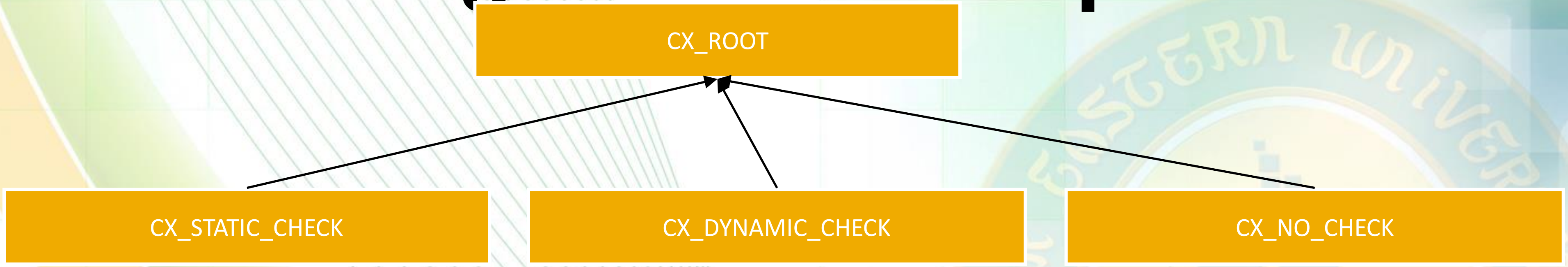
- All exceptions are derived from CX_ROOT
- The order or exception handlers have to be from special to general
- In general it is not advisable to handle CX_ROOT since you do not know any details about the error that occurred.

Declaration of exceptions

- A service (subroutine, function module) defines all exception the caller might have to handle

  `… RAISING cx_a1 cx_a2 cx_flight_not_found`
- Each class mentioned also implies all subclasses
- A service shall only throw exception defined in signature, otherwise exception will be replaced by cx_sy_no_handler exceptions
- The compiler warns if an exception is raised but neither handled nor defined

Categories of exceptions



CX_STATIC_CHECK

Signature statically checked by compiler and at runtime.

User of a service is forced to handle any exception defined as subclass of cx_static_check

CX_DYNAMIC_CHECK

Signature checked only at runtime.

The exception has to be defined in the signature, but the user has the choice to handle it or not.

CX_NO_CHECK

- Exception can occur everywhere
- Exception does not have be defined in signature

Example:
The exception cx_xy_payment_error is probably one, that you always want the calling service to handle.

Example:
The exception cx_sy_zerodivide does not need to be handled by caller, if the caller made sure beforehand, that he does not pass zero-values to service.

Example:
The exception cx_sy_no_more_memory can occur anytime. User does not have to handle event, since he most likely does not have a proper handler.

FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

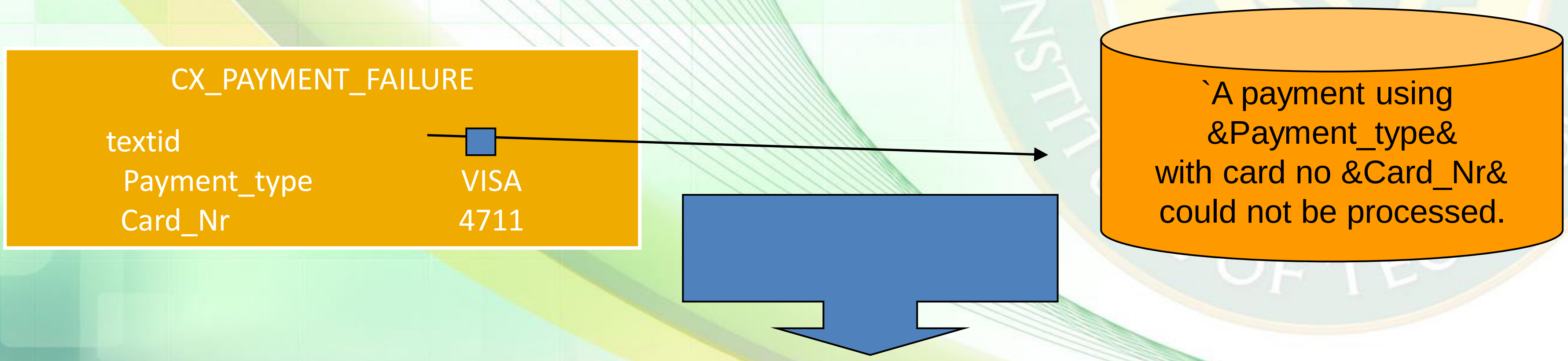
Exception texts

• Each exception has one attribute „textid“

•This attribute points to the textual description of the error

• Method get_text, get_longtext:

–Returns textual description of the description. The textual description is a static text from the class with placeholders for attributes.

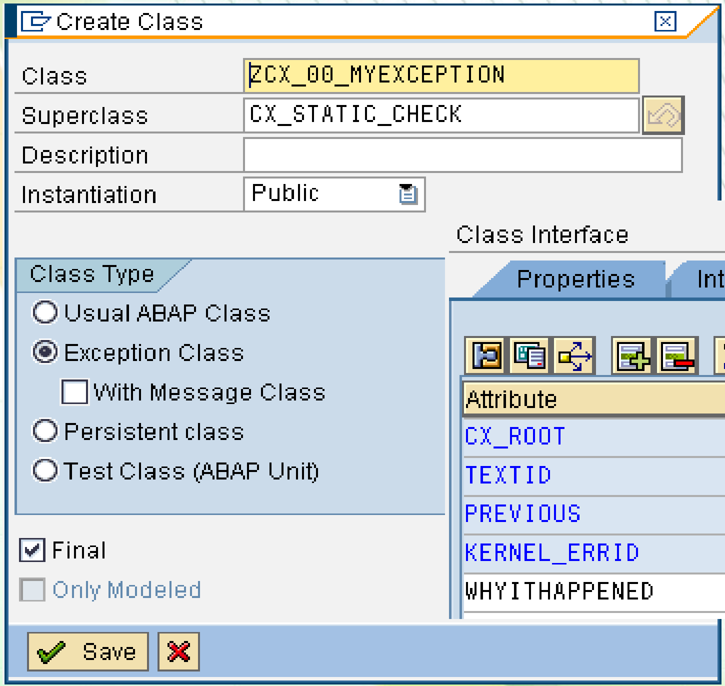

CX_PAYMENT_FAILURE
textid
Payment_type
VISA
Card_Nr
4711
`A payment using &Payment_type& with card no &Card_Nr& could not be processed.

`A payment using VISA with card no 4711 could not be processed.

FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Example 1/4

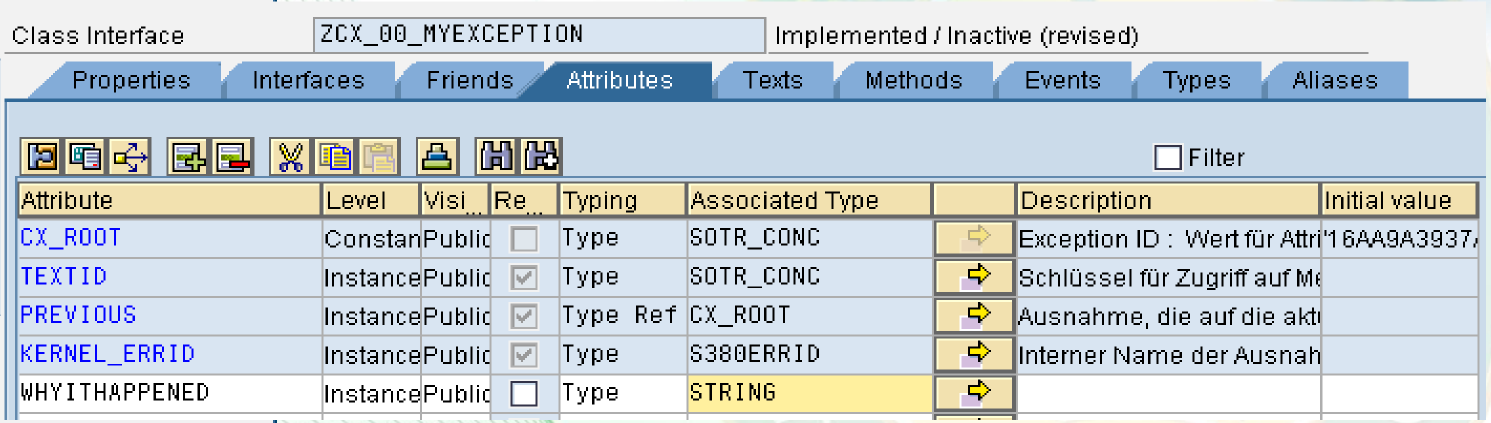
1. Create a new class for the exception


Create Class

Class	ZCX_00_MYEXCEPTION
Superclass	CX_STATIC_CHECK
Description	
Instantiation	Public

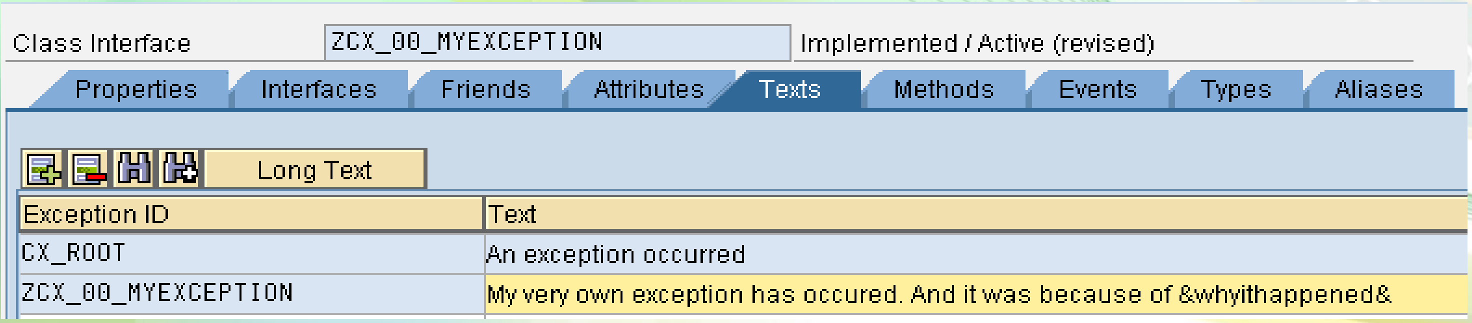
Class Type
- ( ) Usual ABAP Class
- (●) Exception Class
  - [ ] With Message Class
- ( ) Persistent class
- ( ) Test Class (ABAP Unit)

[✓] Final
[ ] Only Modeled


Class Interface ZCX_00_MYEXCEPTION Implemented / Inactive (revised)

Attribute	Level	Visi...	Re...	Typing	Associated Type		Description	Initial value
CX_ROOT	Constan	Public	☐	Type	SOTR_CONC	⇨	Exception ID : Wert für Attri	'16AA9A3937A
TEXTID	Instance	Public	✓	Type	SOTR_CONC	⇨	Schlüssel für Zugriff auf Me	
PREVIOUS	Instance	Public	✓	Type Ref	CX_ROOT	⇨	Ausnahme, die auf die akt	
KERNEL_ERRID	Instance	Public	✓	Type	S380ERRID	⇨	Interner Name der Ausnah	
WHYITHAPPENED	Instance	Public	☐	Type	STRING	⇨		

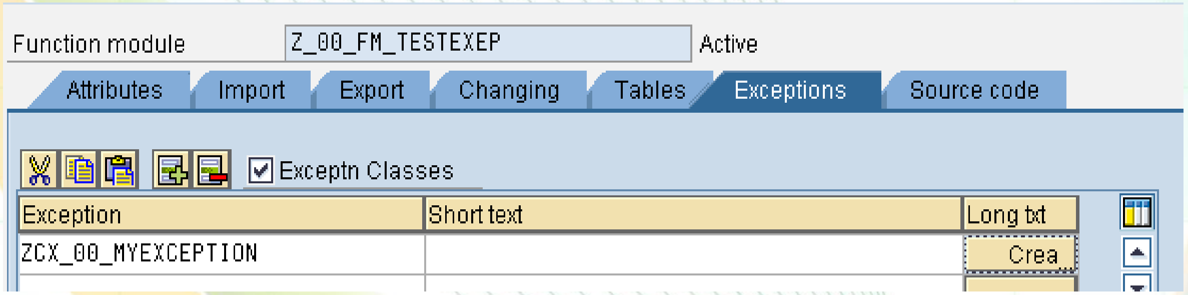
2. Add attributes for more details about the error


Class Interface ZCX_00_MYEXCEPTION Implemented / Active (revised)

Exception ID	Text
CX_ROOT	An exception occurred
ZCX_00_MYEXCEPTION	My very own exception has occured. And it was because of &whyithappened&

3. Set the exception-test using your attributes.

Example 2/4



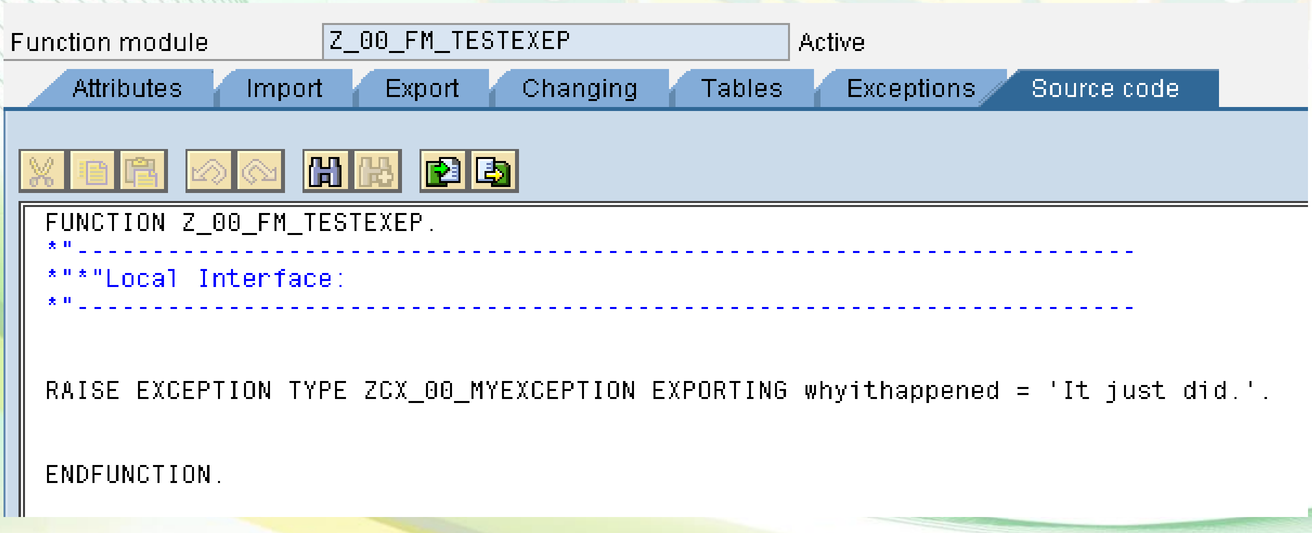
Function module Z_00_FM_TESTEXEP Active

☑ Exceptn Classes

Exception	Short text	Long txt
ZCX_00_MYEXCEPTION		Crea

4. Add your exception to, for example, a function module that can raise the exception.

5. Raise your exception somewhere in the source code providing parameters for your defined attributes.

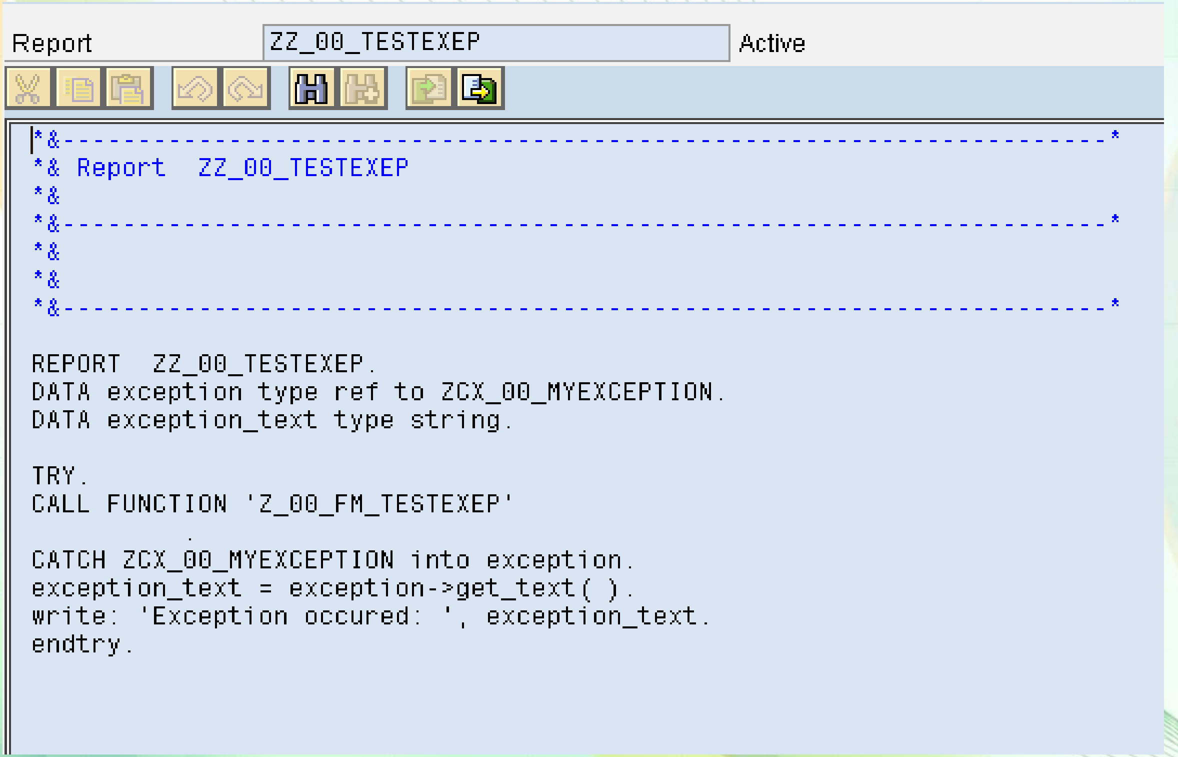


Function module Z_00_FM_TESTEXEP Active

```
FUNCTION Z_00_FM_TESTEXEP.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"----------------------------------------------------------------------


RAISE EXCEPTION TYPE ZCX_00_MYEXCEPTION EXPORTING whyithappened = 'It just did.'.


ENDFUNCTION.
```

FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Example 3/4

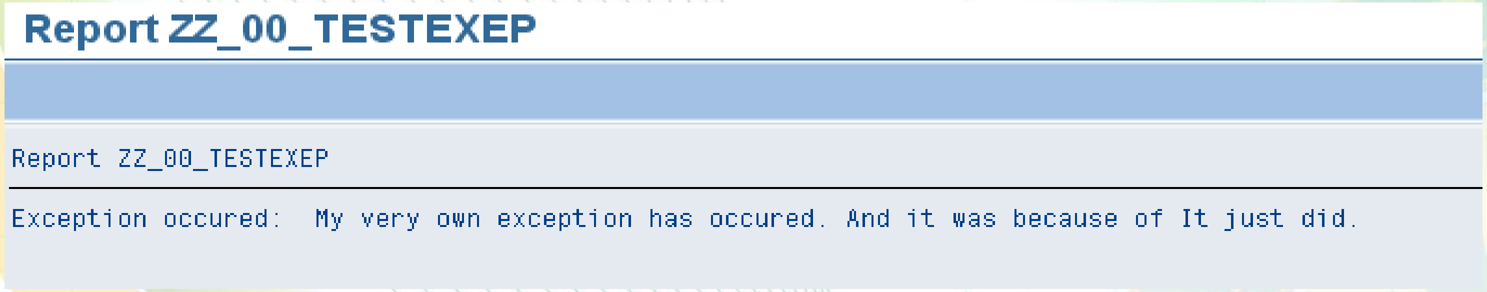

Report ZZ_00_TESTEXEP Active

```
*&---------------------------------------------------------------------*
*& Report  ZZ_00_TESTEXEP
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  ZZ_00_TESTEXEP.
DATA exception type ref to ZCX_00_MYEXCEPTION.
DATA exception_text type string.

TRY.
CALL FUNCTION 'Z_00_FM_TESTEXEP'
          .
CATCH ZCX_00_MYEXCEPTION into exception.
exception_text = exception->get_text( ).
write: 'Exception occured: ', exception_text.
endtry.
```

6. Call the function that can raise your error using a try-clause and a handler for your exception using a catch-clause

FEU INSTITUTE OF TECHNOLOGY
COLLEGE OF ENGINEERING • COLLEGE OF COMPUTER STUDIES

Example 4/4


Report ZZ_00_TESTEXEP

```
Report ZZ_00_TESTEXEP
Exception occured:  My very own exception has occured. And it was because of It just did.
```

7. Test it. Your exception is thrown, handled and displayed to the user.

Outlook

- Other interesting features of exception handling not discussed here include:
  - Cleanup
    - The Cleanup clause can be used to free used resources on unwinding the stack after an error occurs. (for example undo a seat reservation when payment fails)
  - Multiple exception texts
    - One exception class can handle slight variants of an error by providing different exception texts. On raising the exception, the appropriate text is chosen.
  - Resumable exception
    - Exceptions can be marked as resumable. Then the handler can decide whether to abort or continue where the error has occurred. (for example in batch execution the handler writes the error to a log file and continues)