Custom exceptions and exceptions good practices

Agenda

Custom exceptions and exception wrapping

Good exception handling practices

Custom exceptions and exception wrapping

Custom exceptions

You can create your own exceptions if no built-in exception fits your needs

Useful for create application specific exceptions or for security purposes

Always wrap, if possible, the inner exception

Custom exceptions

Create a class that inherits from Exception class or from a derived exception

Include necessary constructors

Exception wrapping

You catch the current exception

You throw a new type of exception

You wrap the current exception in the new exception

The current exception becomes the inner exception for the new exception

Demo

Create custom exceptions in our library

Demo

Throw custom exceptions in the library

Catch the custom exceptions in the console application

Demo

Catch the custom exceptions in the api

Good practices for exception handling

Performance

Handling exception is costly in terms of resources

Use it when necessary

Don’t put try catch statements everywhere or anywhere

When to catch exceptions

When you must handle it (log it, or trigger other actions)

When you want to implement some recovery (file not found for example)

When you want to want to change the exception thrown

When to throw an exception

When a library method argument is invalid

When an API method argument is invalid

When you need to rethrow a custom exception

Don’t do that

```
try
{
    // code
}
catch (Exception ex)
{
    // Exception handling logic
    throw ex; // you lose some info here
}
```

Correct way

```
try
{
    // code
}
catch (Exception ex)
{
    // Exception handling logic
    throw; // no info lost here
}
```

When not to use exceptions

Usage exceptions must be predicted, some checks must be performed to avoid exceptions

Good practices

No program logic in catch blocks

Catch and just rethrow is useless

No silent exception

Clean up resources or close connections in the finally block

On the top of the call stack, in the calling program, handle the exception to display a friendly message and recover

Multiple catch blocks

Catch exceptions from the most specific to the less specific (Exception)

Summary

You can create your own exceptions by deriving the Exception class

You can wrap an exception in another one with the InnerException property

Remember the exception handling good practices