BASTA!

Oliver Sturm

Exception Patterns

Oliver Sturm (@olivers)

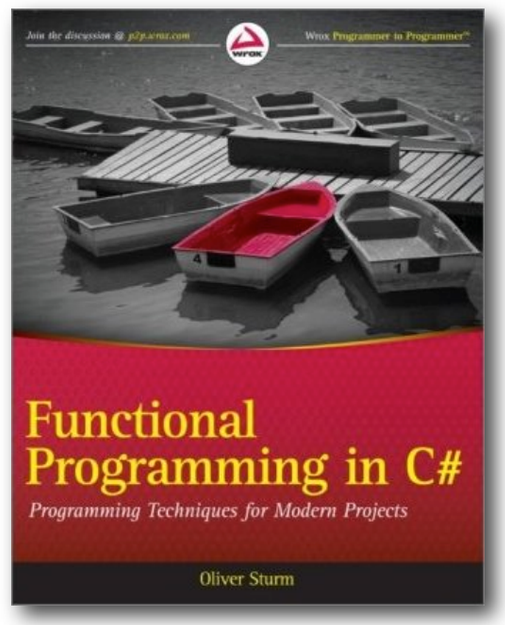
- **Consultant and Trainer, Author**
- **Associate Consultant at thinktecture**

- **.NET Application System Architecture**
  - User Interfaces
  - Data Handling / Data Access Architectures
  - Programming Languages
  - DevExpress Component/Framework Products

- **Microsoft MVP for C#**
- **INETA Europe Speaker**

- **Services: http://www.oliversturm.com**
- **Blog: http://www.sturmnet.org/blog**

- **oliver@oliversturm.com**



Agenda

- Technical Stuff
  - Why exceptions?
  - Catching exceptions
  - Throwing exceptions
  - Which types to use
- The architectural side
  - Guidelines and approaches
  - Anti-patterns

Old Style: Return Codes

- Cons:
  - You had to define them
  - You had to define them *all*
  - Really no fun to do when function calls nest
  - Technically difficult due to single return value
  - In error conditions, they don't tell you very much

- Pros:
  - You had to define them
  - You had to define them *all*

Throwing and Bubbling

- Exceptions are objects that are “thrown”
- They “bubble” up the call stack until they are “caught”
- Language compilers enforce particular types for the exception objects themselves
- Exception types carry lots of useful information about error conditions
  - General stuff like the stack trace
  - Specific info related to the problem at hand

Catching Exceptions

```
try {
   // do something that may throw an exception
   // ...
}
catch (Exception ex) {
   // get here if an exception has been thrown
   // in the block above - handle it using
   // the ‘ex’ variable
}
```

A ‘finally’ block

```
try {
  // do something that may throw an exception
  // ...
}
catch (Exception ex) {
  // get here if an exception has been thrown
  // in the block above - handle it using
  // the ‘ex’ variable
}
finally {
  // you come here in all cases, for instance
  // to do cleanup work
}
```

Demo

Catching exceptions
Using catch/finally
Bubbling in action
Exception Filters considered harmful!
Exceptions need compiler support

Throwing Exceptions

```
throw new SomeException(“Error!”);
```

Demo

Throwing exceptions

Which Exceptions to Use?

- Standard exceptions:
  - ArgumentException
  - ArgumentNullException
  - ArgumentOutOfRangeException
  - InvalidOperationException
  - Others from standard System.XXX namespaces, like FileNotFoundException, …

  - If it's there, you should use it - but read the description and see that it matches!
    - Example description of FileLoadException: "The exception that is thrown when a managed assembly is found but cannot be loaded."

Might want to use code contracts instead in .NET 4.0

Which Exceptions to Use?

- Custom exceptions
  - If there's no matching standard exception
  - If there's a particular way of recovering from the error in question, so having a special type is useful
  - If you have relevant additional information to supply

  - Take care when implementing:
    - [Serializable] and ISerializable
    - Constructors
    - SecurityPermissions
    - Message and/or ToString overrides

Demo

Custom exceptions

Architectural Guidelines

- Catch exceptions in one “place” in your application, on a high level, so that you catch everything
- Accept that exceptions are exceptional. Don’t assume it makes sense to keep the app running.
  - Being able to recover would make the unexpected expected, right?
- Catching, logging and reporting exceptions flawlessly is a difficult task. Consider leaving it to experts.

Demo

Example:
Catching exceptions in a Windows Forms app

Anti-patterns

- Throwing from helper methods
- Using exceptions for flow control
- Catching the wrong way
- Catching and swallowing
- Rethrowing the wrong way
- Catching too much

Demo

Exception anti-patterns

Your own anti-patterns?

- What are your experiences?
- What do you see people do right or wrong with exceptions?

Summary

- Exceptions are a powerful tool
- Easy to use, easy to get wrong, easy to abuse
- You should definitely use exceptions, but use them the right way!

Thank you!

Please feel free to contact me about the content anytime!

oliver@oliversturm.com