SHIELDS UP!
DEFENSIVE CODING IN C#

Presented by Jeremy Clark
www.jeremybytes.com

Goals

Robust and **Effective** Code

Understand **Common Problems**

Code for the **Real World**

THROWING EXCEPTIONS

- Throw specific exceptions, such as
  - ArgumentNullException
  - NullReferenceException
  - AccessViolationException

- Avoid throwing “Exception”

THROWING “EXCEPTION”

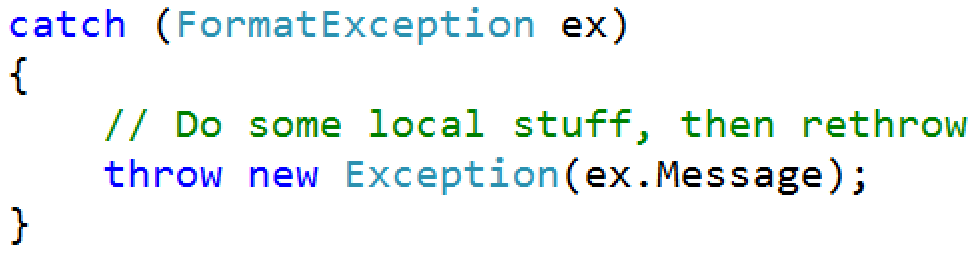
THROWING EXCEPTIONS

- Throw specific exceptions, such as
  - ArgumentNullException
  - NullReferenceException
  - AccessViolationException
- Avoid throwing “Exception”
- Be aware that exceptions are “expensive”

CATCHING EXCEPTIONS

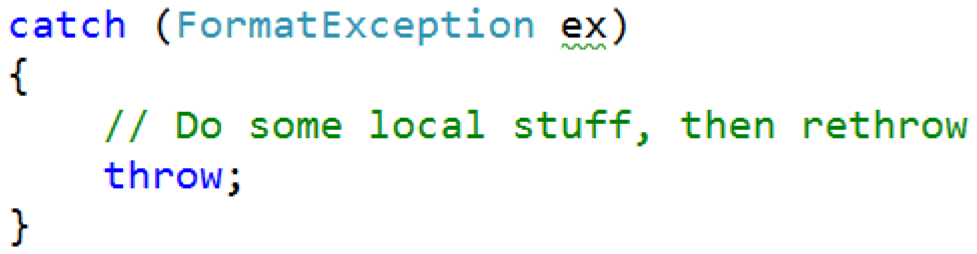
- Use try blocks where exceptions could occur
  - try/catch where you can handle the exception
  - try/finally where you cannot handle the exception
- Catch specific exceptions
- Only catch an exception if you an do something with it
- Have a global exception handler (for everything else)

RETHROWING EXCEPTIONS



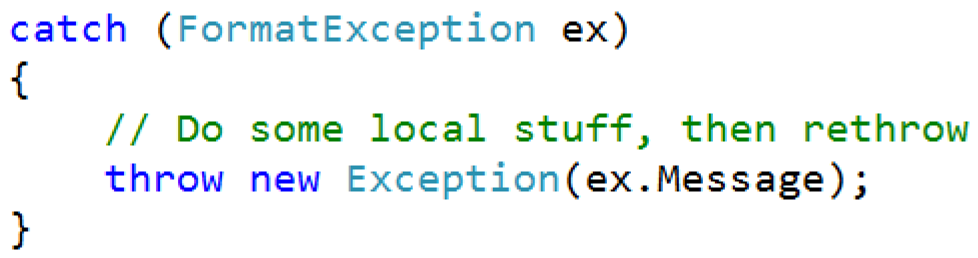
```
catch (FormatException ex)
{
    // Do some local stuff, then rethrow
    throw new Exception(ex.Message);
}
```

VS



```
catch (FormatException ex)
{
    // Do some local stuff, then rethrow
    throw;
}
```

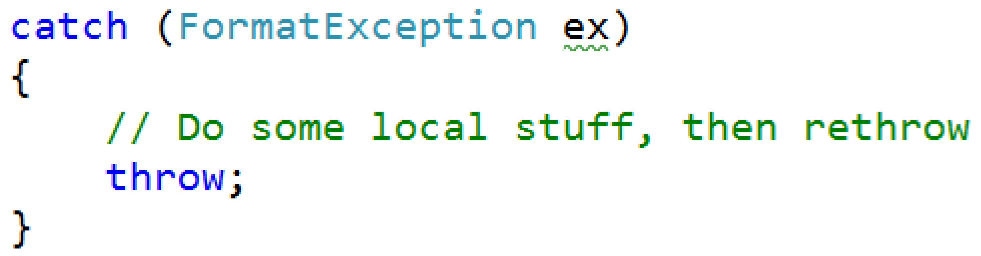
RETHROWING EXCEPTIONS



```
catch (FormatException ex)
{
    // Do some local stuff, then rethrow
    throw new Exception(ex.Message);
}
```

- Creates a new Exception object
- Resets the stack trace
- We don’t know where the original exception was generated

RETHROWING EXCEPTIONS



```
catch (FormatException ex)
{
    // Do some local stuff, then rethrow
    throw;
}
```

- Rethrows the same exception object
- The stack trace (and other properties) are retained
- We can look at the stack trace to find the exception origin

All Input is Evil

INPUT VALIDATION

- Validate parameters on public methods
  - Null checking
  - IsNullOrWhiteSpace() for strings
  - Range checking
  - IsDefined() for enumerations

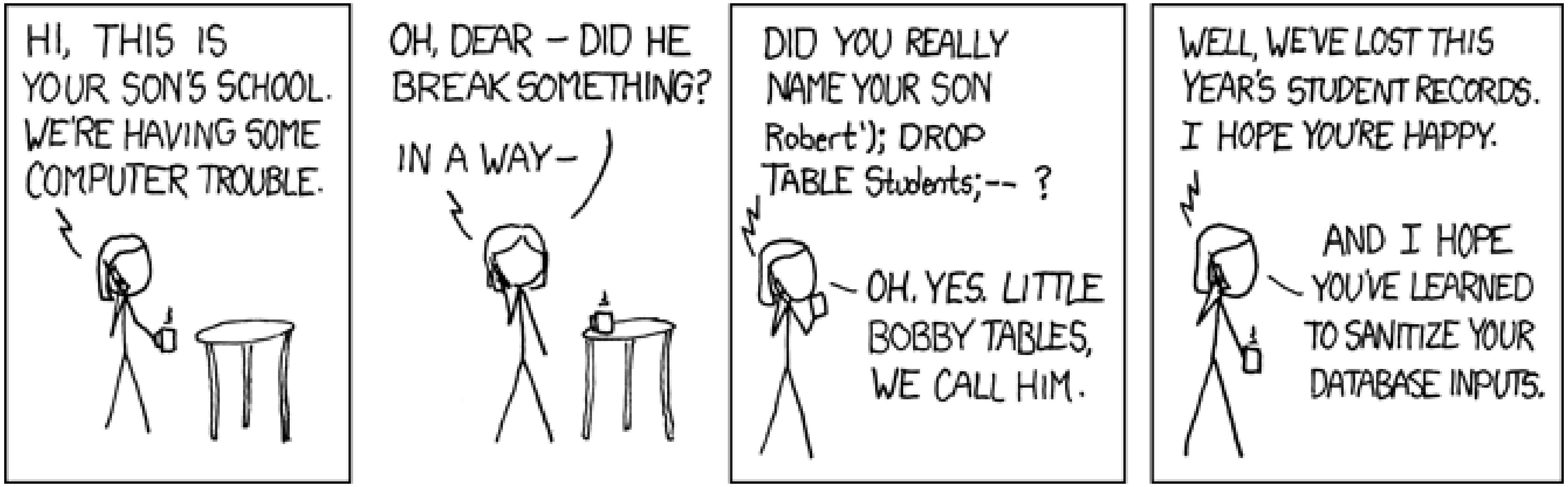
- Parse vs. TryParse
  - Parse will throw an exception on failure
  - TryParse returns "false" on failure (no exception)

DEMO:
INPUT VALIDATION & EXCEPTION HANDLING

SQL INJECTION

User input that is executed as a SQL command.

DEMO:
SQL INJECTION


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

http://xkcd.com/327/

PREVENTING SQL INJECTION

- Parameterized Queries
- Stored Procedures (which are parameterized)
- ORM Frameworks are built to prevent SQL injection
  - Entity Framework
  - Nhibernate
  - Many others

DEMO: PARAMETERIZED SQL

HACKABLE URIS

- Entry Points
  - URL query strings
  - "Pretty" URLs with parameters (like ASP.NET MVC)
  - REST Services
  - WebAPI
  - Other technologies that use HTTP as the main form of passing parameters

DEMO: HACKABLE URIS

SECURING URIS

- User Validation
  - For user-specific data, make sure the data user matches the requesting user.
- Authorization
  - For other controlled data, check security settings to make sure the user is authorized

DEMO:
SECURING URIS

UNIT TESTING

Unit Tests are proof that your code actually does what you think it does.

UNIT TESTING

- Testing small pieces of code
  - Usually on the method level

- Testing in isolation
  - Eliminate outside interactions that might break the test
  - Reduce the number of objects needed to run the test

- Note: We still need Integration Testing
  - Testing that the pieces all work together

OTHER TOPICS

- IDisposable
  - If an object implements IDisposable, make sure you call "Dispose()" or wrap the object in a "using" statement
- Event Handlers
  - Disconnect event handlers when you're done with them
  - A connected event handler prevents Garbage Collection
  - Alternately, use a weak-reference event handling process, such as the EventAggregator class from Microsoft p&p

KEEP
CALM
AND
RAISE
SHIELDS

SHIELDS UP!
DEFENSIVE CODING IN C#

Presented by Jeremy Clark
www.jeremybytes.com