Ruby 101

Intermediate Ruby: Exception Handling

Intermediate Ruby:

What happens when....

Users do the unexpected?
programs do odd things?

Exceptions

Exceptions are:
- Errors in the processing
- Defined in advance by either Ruby, or the programmer

Exceptions can be dealt with!

Intermediate Ruby:

Exceptions and the Begin/Rescue/End program block.

```
begin           -> Specifies an exception handling block
    ...         -> Code which may raise an exception
rescue          -> If we ge an exception, execute the following
    ...
end
```

Intermediate Ruby:

Retry:

```
begin
    ...
rescue
    ...
    retry -> Goes back to the begining of the block
end
```

Intermediate Ruby:

Ensure:

```
begin
    ...
rescue
    ...
ensure
    ....  -> No matter what, run this code.
end
```

Intermediate Ruby:

Else:

```
begin
    ...
rescue
    ...
else
    ... -> Run this only if no exceptions were raised.
ensure
    ...
end
```

Intermediate Ruby:

specify which exception to rescue:

```
rescue MySillyError
```

Multiple different exceptions:

```
rescue ErrorA
    ...
rescue ErrorB
    ...
```

Intermediate Ruby:

Pass the exception along, but still do something:

```
begin
    ...
rescue
    ...          -> Maybe inform the user what went wrong
    raise
end
```

Intermediate Ruby:

Access the exception object:

```
begin
    ...
rescue => the_exception
    puts the_exception.backtrace
    ...
end
```

Intermediate Ruby:

Raising our own exceptions:

raise *string*

e.g.

```
raise "My, oh my, what a terrible thing!"

class MyException < RuntimeError
end

raise MyException, "This was a bad day", ["line 5 of good_days.rb", "line 10 of daily_notes.rb"]
```

Intermediate Ruby:

How do I know what Exception?

ruby-doc.org documentation
programming experience
extensive code testing

User input was blank/nil?
No permission to access something
End of a file was reached

In our program:
read_nonblock raises *IO::WaitReadable*

Intermediate Ruby:

controller.rb

```
begin
    while next_chars = $stdin.read_nonblock(10) do
        user_input = "#{user_input}#{next_chars}"
    end
rescue IO::WaitReadable
    # No code here, since the error just means we got
    # all the data
end
```