Ruby ve Rails'de Exception Kullanımı

Erek Göktürk

Özgür Yazılım A.Ş.
www.ozguryazilim.com.tr

Niye bu konu?

- Betik dili kullanımı giderek artıyor.
- Modern betik dilleri modern programlama dilleri kavramlarını gayet güzel içeriyorlar.
- Betik dilleri çalakalem yazmaya pek uygun.
- Programcılar var olan modern teknikleri kullanmaktan nedense kaçıyorlar.
- Bu tekniklerden biri de Exception kullanımı.
- Uyarı: Yerinde ve kararınca kullanmazsanız, işinizi kolaylaştırmak yerine kodunuzu gerçekten karıştırabilirsiniz.

Exception nedir?

- Bir akış kontrol yöntemi

```
begin
  ... # e yaratabilecek bir kod
catch Exception => e
  ... # e'yi işle
end


def fonksiyonum
  ... # e yaratabilecek bir kod
catch Exception => e
  ... # e'yi işle
end
```

Exception niye kullanılır?

- İyi program tasarımı, programın küçük ve amaca odaklı fonksiyonlara bölümlenmesini gerektirir.
- Hatanın ne olduğunu en iyi işi yapmaya çalışan bilir (en altlarda)
- Hatanın nasıl işleneceğini genelde işin yapılmasını isteyen bilir (daha yukarılarda)
- Aradaki aktarım için Exception'lar hata durumları için alternatif kontrol akış yolları sağlarlar.

Exception niye kullanılır?

```
def a
  ...
  a_helper
  # a_helper başarılı oldu mu?
  ...
end

def a_helper
  ...
  a_helper_helper
  ...
end

def a_helper_helper
  ...
  # burada hata dönmemiz gerek
  ...
end
```

Exception niye kullanılır?

```
def a
  ...
  sonuc = a_helper
  if sonuc
    # a_helper başarılı oldu
  else
    # a_helper başarısız oldu
  end
  ...
end

def a_helper
  ...
  sonuc = a_helper_helper
  return nil unless sonuc
  ...
end

def a_helper_helper
  ...
  # burada hata dönmemiz gerek
  return nil     # ya nil de dönebilmemiz gerekiyorsa?
  ...
end
```

Exception niye kullanılır?

```
def a
  ...
  begin
    a_helper
  rescue Exception => e
    # a_helper başarılı olamadı
  end
  # a_helper başarılı oldu
  ...
end

def a_helper
  ...
  a_helper_helper
  ...
end

def a_helper_helper
  ...
  raise 'hell' if not_good
  ...
end
```

```
def a
  ...
  a_helper
  # a_helper başarılı oldu
  ...
rescue Exception => e
  # a_helper başarılı olamadı
end
```

Ruby'de Exception

- BasicObject
  - Exception
    - IRB::Abort
    - NoMemoryError
    - ScriptError
      - LoadError
        - Gem::LoadError
      - NotImplementedError
      - SyntaxError
    - SecurityError
    - SignalException
      - Interrupt
    - StandardError
      - ArgumentError
      - EncodingError
        - Encoding::CompatibilityError
        - Encoding::ConverterNotFoundError
        - Encoding::InvalidByteSequenceError
        - Encoding::UndefinedConversionError
      - Exception2MessageMapper::ErrNotRegisteredException
      - FiberError
      - IOError
        - EOFError
      - IRB::CantChangeBinding
      - IRB::CantReturnToNormalMode
      - IRB::CantShiftToMultiIrbMode
      - IRB::IllegalParameter
      - IRB::IrbAlreadyDead
      - IRB::IrbSwitchedToCurrentThread
      - IRB::NoSuchJob
      - IRB::NotImplementedError
      - IRB::Notifier::ErrUndefinedNotifier
      - IRB::Notifier::ErrUnrecognizedLevel
      - IRB::SLex::ErrNodeAlreadyExists
      - IRB::SLex::ErrNodeNothing
      - IRB::UndefinedPromptMode
      - IRB::UnrecognizedSwitch
      - IndexError
        - KeyError
        - StopIteration
      - LocalJumpError
      - Math::DomainError
      - NameError
        - NoMethodError
      - RangeError
        - FloatDomainError

Ruby'de Exception

- RegexpError
- RubyLex::AlreadyDefinedToken
- RubyLex::SyntaxError
- RubyLex::TerminateLineInput
- RubyLex::TkReading2TokenDuplicateError
- RubyLex::TkReading2TokenNoKey
- RubyLex::TkSymbol2TokenNoKey
- RuntimeError
- Gem::Exception
- Gem::CommandLineError
- Gem::DependencyError
- Gem::DependencyRemovalException
- Gem::DocumentError
- Gem::EndOfYAMLException
- Gem::FilePermissionError
- Gem::FormatException
- Gem::GemNotFoundException
- Gem::GemNotInHomeException
- Gem::InstallError
- Gem::InvalidSpecificationException
- Gem::OperationNotSupportedError
- Gem::RemoteError
- Gem::RemoteInstallationCancelled
- Gem::RemoteInstallationSkipped
- Gem::RemoteSourceException
- Gem::VerificationError
- SystemCallError
- ThreadError
- TypeError
- ZeroDivisionError
- SystemExit
- Gem::SystemExitException
- SystemStackError
- fatal

Rails'de Exception

- Rails exception yönetmek için kendi eklentisi, ve kendi exception tanımları ile geliyor.

```
class ApplicationController < ActionController::Base
  rescue_from User::NotAuthorized, :with => :deny_access # self defined exception
  rescue_from ActiveRecord::RecordInvalid, :with => :show_errors

  rescue_from 'MyAppError::Base' do |exception|
    render :xml => exception, :status => 500
  end

  protected
    def deny_access
      ...
    end

    def show_errors(exception)
      exception.record.new_record? ? ...
    end
end
```

Ruby'de neden rescue_from yok?

- Yazması çok mu basit? Hayır.
  - Bir nesne veya sınıfın herhangi bir fonksiyonunda oluşan bir Exception'ı yakalayıp yönlendirebilmesi gerekiyor.
  - Ruby'de call stack'te araya girmek için bir yöntem yok (çağrı zamanında fonksiyon sarmalamak mümkün değil)
- Olmasına gerek mi yok? Tartışılır.

Exceptionist (çok yakında)

- Bir de şu durumlar var:
  - Bir sınıf metodunda ortaya çıkan Exception'ın belli bir metot ile işleneceğini belirtmek
  - Bir nesne metodunda ortaya çıkan Exception'ın belli bir metot ile işleneceğini belirtmek
- Epeyce kurcaladıktan sonra, ruby izin vermese de bu işi yapmanın mümkün olduğunu keşfettim galiba :)

https://github.com/mdasheg/exceptionist

Exceptionist (çok yakında)

```
# class A
#   # Two alternative definitions of a handler for a given exception.
#   # my_method is an instance method, and :instance_method is a keyword.
#   rescue_exception MyException, :in => :my_method, :with => :handler_method
#   rescue_exception MyException, :in => [:instance_method, :my_method],
#     :with => :handler_method
#
#   # definition of a specific handler for a given exception.
#   # my_method is a singleton method
#   rescue_exception MyException, :in => :my_method, :with => :handler_method
#   rescue_exception MyException, :in => [:singleton_method, :my_method],
#     :with => :handler_method
#
#   # definition of a catch-all exception handler for a given method
#   rescue_exceptions :in => my_method, :with => :another_handler_method
#
#   # definition of a catch-all exception handler for all instance methods
#   rescue_exceptions :with => :handler_for_all_methods
# end
```

Exceptionist – Niye çok yakında?

- Yorumlayıcıya yakın bir düzeyde ve her nesne/sınıfa burnunu sokan bir gem. Doğru ve performanslı çalıştığından emin olmak gerekiyor.
- Testlerinin yazımı devam ediyor.
- Genişletilebileceği bazı yönler var.

Sorular ve sohbet...