An Introduction of Interator Pattern with Ruby

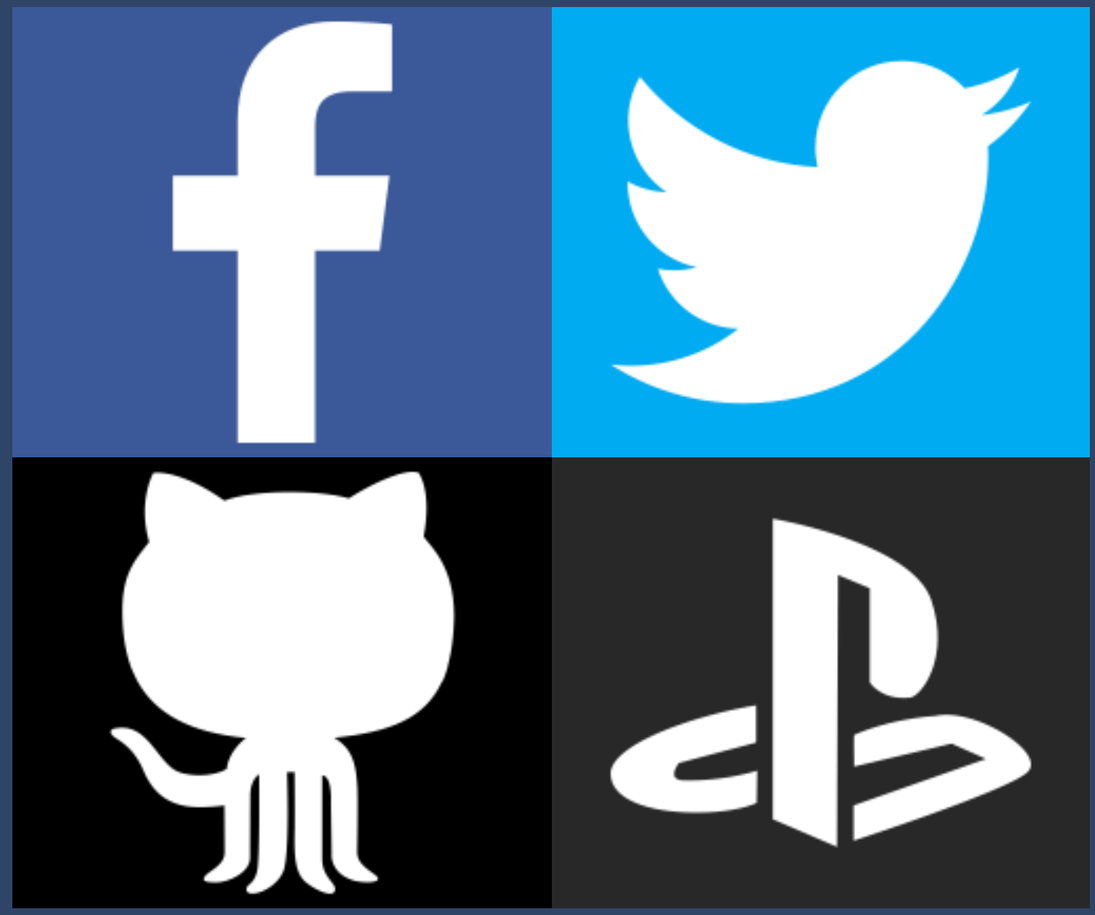
@ ConFoo Montreal 2018-03-07
by Jian Weihang

Bonjour!

簡煒航

Jian Weihang

@tonytonyjan



Taiwan


Ireland
Kingdom
Poland
Belarus
Germany
Ukraine
Austria
France
Romania
Kazakhstan
Mongolia
Italy
Spain
Uzbekistan
Kyrgyzstan
Portugal
Greece
Turkey
Turkmenistan
China
South Korea
Syria
Tunisia
Iraq
Iran
Afghanistan
Morocco
Pakistan
East China Sea
Algeria
Libya
Egypt
Nepal
Western Sahara
Saudi Arabia
India
Myanmar (Burma)
Oman
Mauritania
Mali
Niger
Sudan
Yemen
Thailand
Chad
Gulf of Aden
Arabian Sea
Bay of Bengal
Vietnam
Philippines
Burkina Faso
Guinea
Nigeria
Ethiopia
Gulf of Thailand
Ghana
South Sudan
Laccadive Sea
Somalia
Malaysia
Gulf of Guinea
Kenya
Gabon
Indonesia
DR Congo
Banda Sea
Tanzania
Angola
Zambia
Mozambique
Namibia
Zimbabwe
Madagascar
Botswana
Indian Ocean
South Africa
North Atlantic Ocean
South Atlantic Ocean
United States
Mexico
Gulf of Mexico
Cuba
Puerto Rico
Guatemala
Caribbean Sea
Nicaragua
Venezuela
Guyana
Colombia
Suriname
Ecuador
Brazil
Peru
Bolivia
Paraguay
Chile
Uruguay
Argentina


Kingdom
Ireland
Belarus
Poland
Germany
Ukraine
Kazakhstan
Mongolia
Austria
France
Romania
Italy
Spain
Uzbekistan
Kyrgyzstan
Greece
Turkey
Turkmenistan
Portugal
China
South Korea
Syria
Tunisia
Iraq
Afghanistan
Iran
Morocco
Pakistan
East China Sea
Algeria
Libya
Egypt
Nepal
Western Sahara
Saudi Arabia
India
Myanmar (Burma)
Oman
Mauritania
Mali
Niger
Sudan
Yemen
Thailand
Chad
Gulf of Aden
Arabian Sea
Bay of Bengal
Burkina Faso
Guinea
Nigeria
Vietnam
Philippines
Ethiopia
Gulf of Thailand
Ghana
South Sudan
Laccadive Sea
Somalia
Gulf of Guinea
Malaysia
Kenya
Gabon
Indonesia
DR Congo
Banda Sea
Tanzania
Angola
Zambia
Mozambique
Namibia
Zimbabwe
Madagascar
Botswana
South Africa
Indian Ocean
North Atlantic Ocean
South Atlantic Ocean
United States
Mexico
Gulf of Mexico
Cuba
Puerto Rico
Guatemala
Caribbean Sea
Nicaragua
Venezuela
Guyana
Colombia
Suriname
Ecuador
Brazil
Peru
Bolivia
Paraguay
Chile
Uruguay
Argentina


Ireland
Kingdom
Poland
Belarus
Germany
Ukraine
Austria
France
Romania
Italy
Spain
Greece
Turkey
Portugal
Kazakhstan
Mongolia
Uzbekistan
Kyrgyzstan
Turkmenistan
China
South Korea
Syria
Iraq
Iran
Afghanistan
Pakistan
Nepal
India
Myanmar (Burma)
Thailand
Vietnam
Philippines
Malaysia
Indonesia
Tunisia
Morocco
Algeria
Libya
Egypt
Western Sahara
Saudi Arabia
Oman
Yemen
Mauritania
Mali
Niger
Chad
Sudan
Burkina Faso
Guinea
Nigeria
Ghana
Ethiopia
South Sudan
Somalia
Kenya
Gabon
DR Congo
Tanzania
Angola
Zambia
Mozambique
Namibia
Zimbabwe
Botswana
Madagascar
South Africa
United States
Mexico
Gulf of Mexico
Cuba
Puerto Rico
Guatemala
Nicaragua
Caribbean Sea
Venezuela
Guyana
Suriname
Colombia
Ecuador
Peru
Brazil
Bolivia
Paraguay
Chile
Uruguay
Argentina
North Atlantic Ocean
South Atlantic Ocean
Gulf of Guinea
Gulf of Aden
Arabian Sea
Laccadive Sea
Bay of Bengal
Gulf of Thailand
East China Sea
Philippine Sea
Banda Sea
Indian Ocean

```
$ gem install taiwan
```

Tech Leader

ROUND **TAIWAN** ROUND

Ruby Developer
since 2010

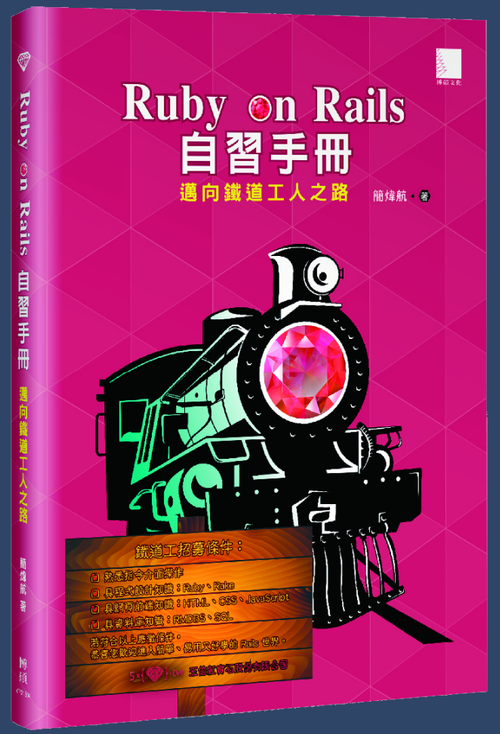
Maintainer
of exif and jaro_winkler

```
└─λ gem query -red exif jaro_winkler | grep --color -E '^exif|^jaro_winkler|Jian Weihang|$'
exif (2.2.0)
    Author: Jian Weihang
    Homepage: https://github.com/tonytonyjan/exif

    Ruby EXIF reader written in C extension.
jaro_winkler (1.5.0, 1.4.0)
    Platforms:
        1.4.0: java
        1.5.0: ruby
    Author: Jian Weihang
    Homepage: https://github.com/tonytonyjan/jaro_winkler

    An implementation of Jaro-Winkler distance algorithm written \ in C
    extension which supports any kind of string encoding.
```

Published a book in 2015



An Introduction of Interator Pattern in Ruby

Outline


RAILS

How to Help Controllers Lose Weight?

The Controller Is a Translator

Good Practice

```
class PostsController < ApplicationController
  def create
    @post = Post.create(post_params)
    redirect_to @post
  rescue
    render :new
  end
end
```

Bad Practice

```
class PostsController < ApplicationController
  def create
    @post = Post.create(post_params)
    @post.send_notification
    redirect_to @post
  rescue
    render :new
  end
end
```

Fat Models and Skinny Controllers

Single Responsibility Principle

```ruby
# Rails 5.1.5
class Post < ActiveRecord::Base
end

Post.new.public_methods.size # => 404
```

```ruby
class AlmightyGod < ActiveRecord::Base
  # hundreds of methods
end
```

```ruby
class Cart < ApplicationModel
  def add_product(product)
  end
end
```

vs.

```ruby
class Product < ApplicationModel
  def add_to_cart(cart)
  end
end
```

```ruby
class Course
  def add_student(student)
  end
end
```

vs.

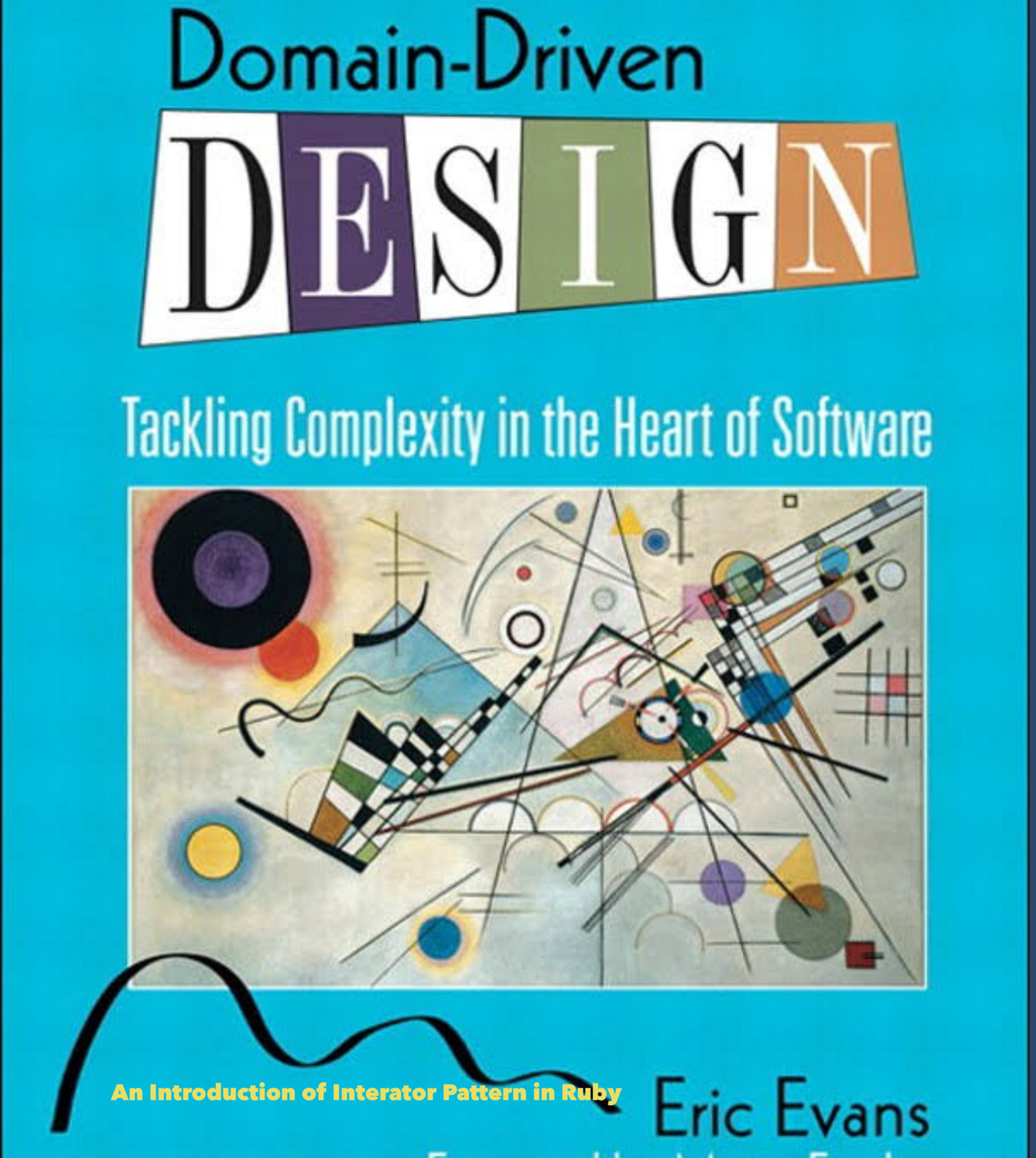
```ruby
class Student
  def join_course(course)
  end
end
```

Model is neither a class nor object.

Model is a layer.

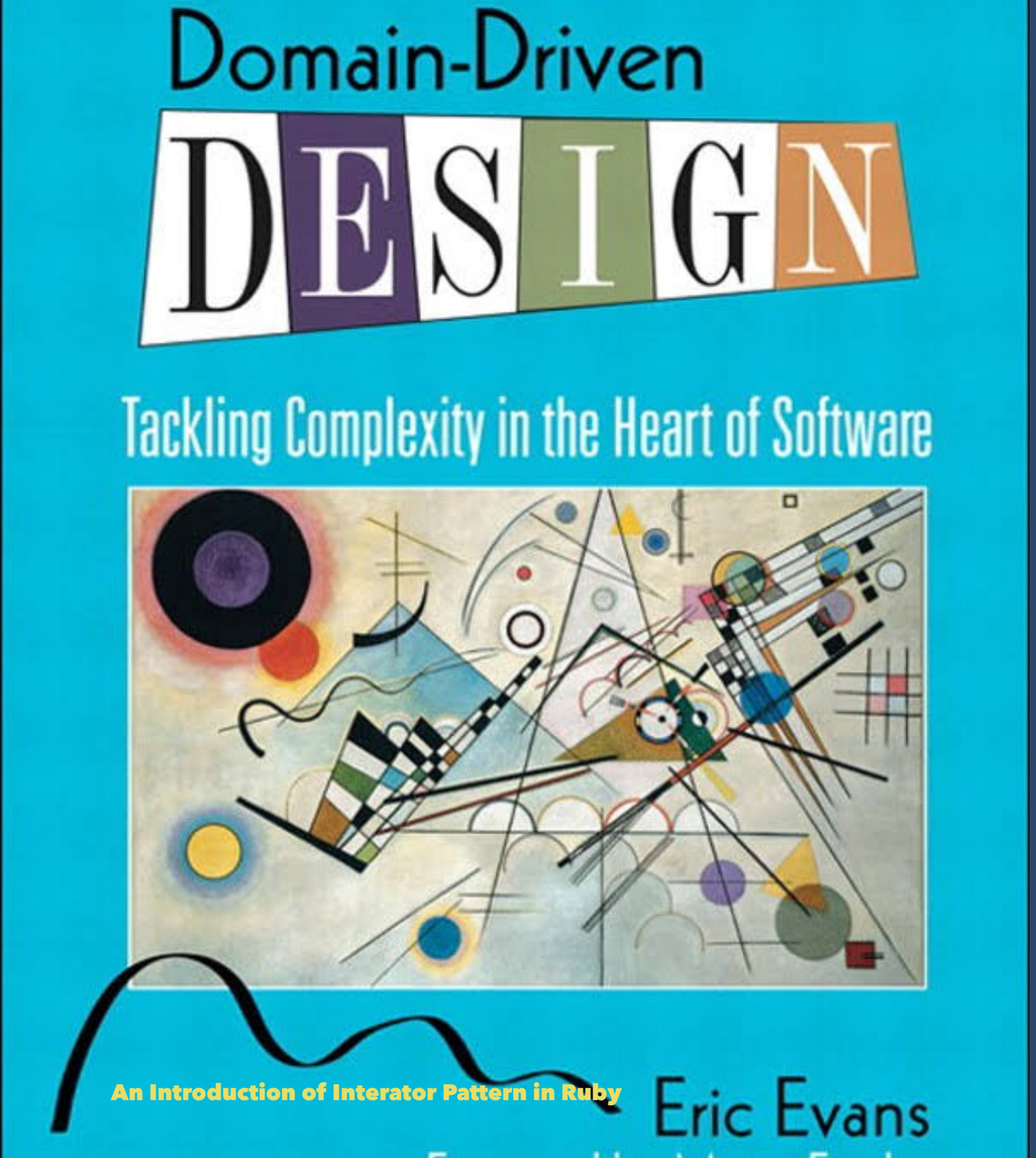
Separation of Concerns

- Presentation Layer
  - views
  - controllers
- Model Layer
  - models



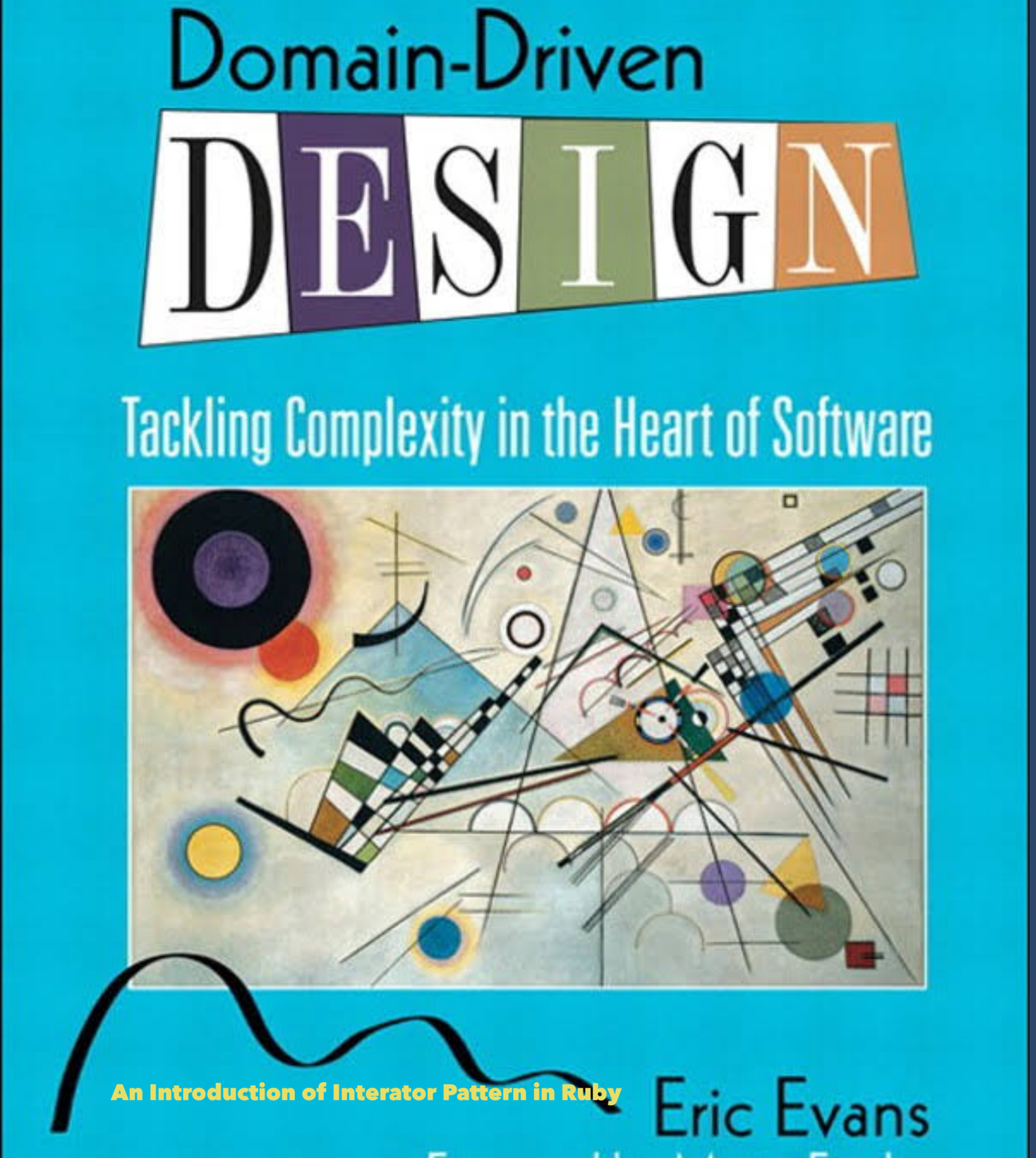
Model Layer

- Domain Objects
- Storage Abstractions
- Services



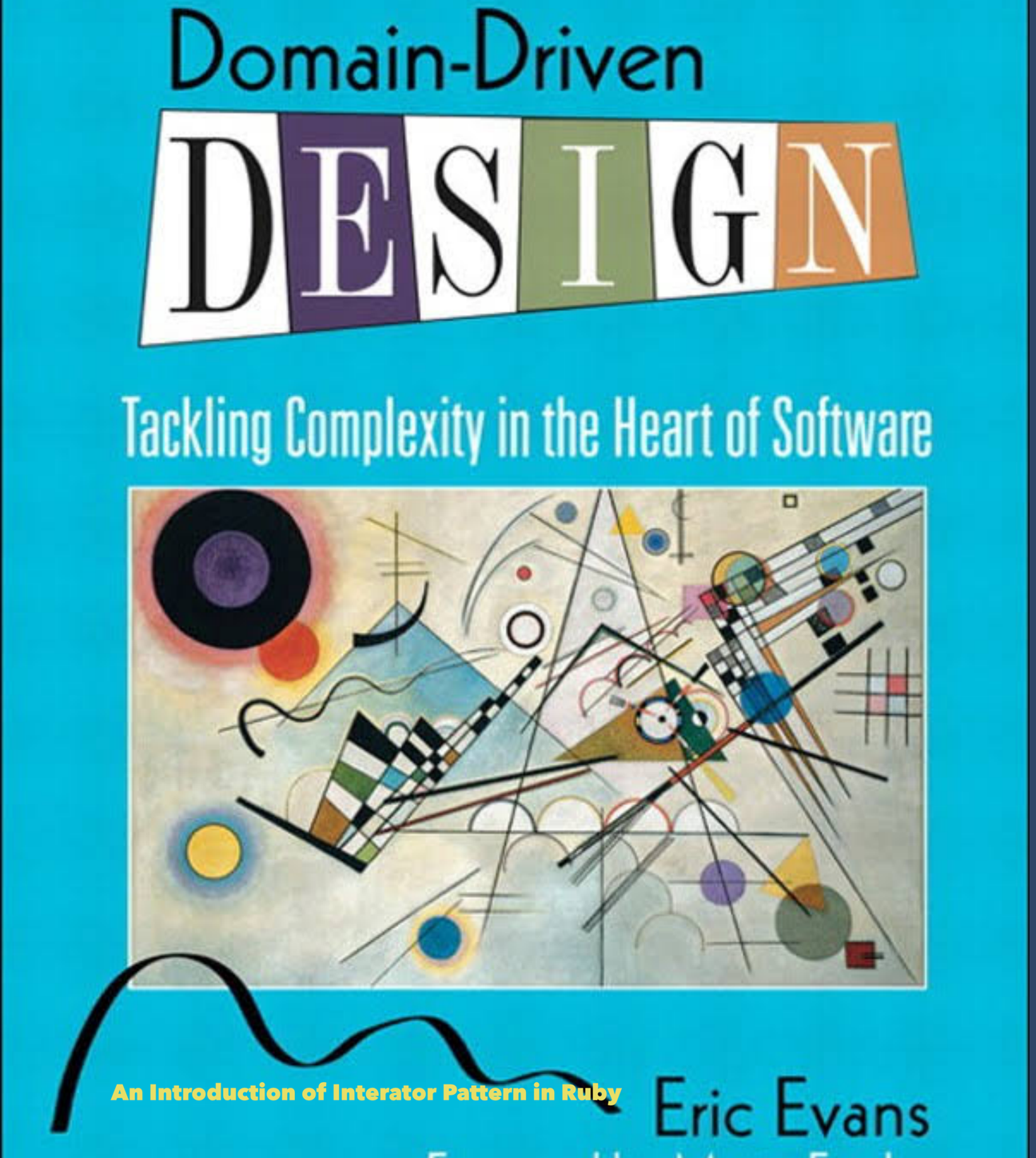
Model Layer

- **Domain Objects**
- Storage Abstractions
- Services



Model Layer

- Domain Objects
- **Storage Abstractions**
- Services



Model Layer

- Domain Objects
- Storage Abstractions
- **Services**

Model Layer

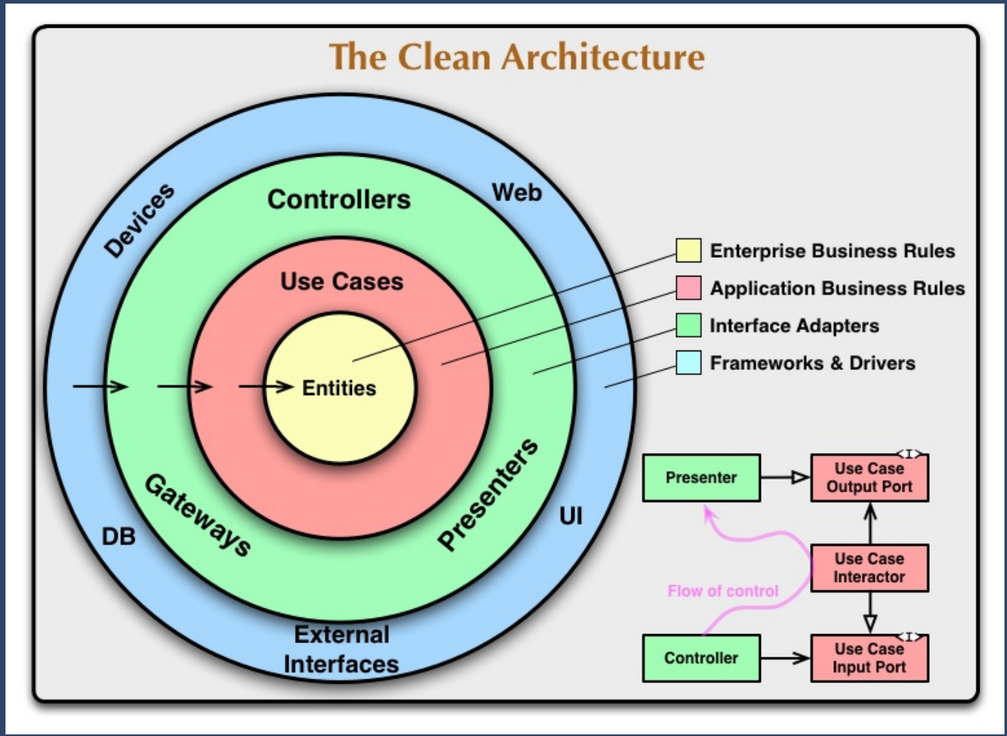
- ActiveRecord::Base
  - Domain Objects
  - Storage Abstractions
- Services
  - ???

What about the Implementation 

Service Object

- a.k.a.
  - Interactor
  - Use Case

Clean Architecture



https://goo.gl/ihgHGB

Minimal Implementation

```ruby
# app/{services|interactors|use_cases}
class AddToCart
  attr_reader :errors

  def initialize(cart, product)
    @cart, @product = cart, product
    @errors = []
  end

  def perform
    # manipulate @cart, @product and @errors ...
  end

  def success?; @errors.empty? end
end
```

Usage

```
class CartController < ApplicationController
  def add_to_cart
    product = Product.find(params[:product_id])
    add_to_cart = AddToCart.new(current_cart, product)
    add_to_cart.perform
    if add_to_cart.successs?
      # ...
    else
      # ...
    end
  end
end
```

Advantages

- Skinny Controller
- Easier to Trace Code
- Easier to Write Test

Advantages

- **Skinny Controller**
- Easier to Trace Code
- Easier to Write Test

Advantages

- Skinny Controller
- **Easier to Trace Code**
- Easier to Write Test

Advantages

- Skinny Controller
- Easier to Trace Code
- **Easier to Write Test**

```ruby
class AddToCart
  def initialize(cart, product)
    @cart, @product = cart, product
  end

  def perform
    @cart.line_items.push(@product)
  end
end
```

gem 'rspec-mocks'

```
describe AddToCart
  it 'works' do
    line_items = double('line_items')
    cart = double('cart', line_items: line_items)
    product = double('product')

    expect(line_items).to receive(:push).once { product }
    # expect cart.line_items.push(product)

    add_to_cart = AddToCart.new(cart, product)
    add_to_cart.perform
  end
end
```

Abstraction

```
class Interactor
  attr_reader :errors

  def self.perform(*args)
    new(*args).tap { |interactor| interactor.perform }
  end

  def perform
    raise NotImplementedError
  end

  def success?
    @errors.empty?
  end
end
```

Usage

```
class AddToCart < Interactor
  def initialize(cart, product)
    @cart = cart
    @product = product
  end

  def perform
    # logic here.
  end
end
```

Usage

```
class CartController < ApplicationController
  def add_to_cart
    product = Product.find(params[:product_id])
    add_to_cart = AddToCart.perform(current_cart, product)
    if add_to_cart.successs?
      # ...
    else
      # ...
    end
  end
end
```

Third-party Gems

gem "mutations"

```ruby
class UserSignup < Mutations::Command
  required do
    string :email, matches: EMAIL_REGEX
    string :name
  end

  optional do
    boolean :newsletter_subscribe
  end

  def execute
    # logic
  end
end

UserSignup.run(params[:user])
```

gem "interactor"

```
class AuthenticateUser
  include Interactor

  def call
    if user = User.authenticate(context.email, context.password)
      context.user = user
      context.token = user.secret_token
    else
      context.fail!(message: "authenticate_user.failure")
    end
  end
end

AuthenticateUser.call(params)
```

FYI

name	stars	forks	issues	closed
mutation	1,127	69	54	79%
interactor	1,919	110	76	87%

gem "interactor2"

```ruby
class AddToCart < Interactor2
  attr_reader :line_item, :cart # should be any attribute you want to expose

  def initialize(product, cart)
    @cart = cart
    @line_item = cart.line_items.new(product: product)
  end

  def perform
    unless @line_item.save
      fail! 'oops'
    end
  end
end
```

gem "interactor2"

```ruby
class AddToCart < Interactor2
  include ActiveModel::Validations
  attr_reader :line_item, :cart # should be any attribute you want to expose

  validates :product, :cart, presence: true

  def initialize(product, cart)
    @cart = cart
    @line_item = cart.line_items.new(product: product)
  end

  def perform
    unless @line_item.save
      fail! 'oops'
    end
  end
end
```

https://github.com/tonytonyjan/interactor2/

Conclusion

- Consistent Interface
- Self-explaining
- Context Encapsulation
- Decoupling your domain from your framework

Conclusion

- **Consistent Interface**
- Self-explaining
- Context Encapsulation
- Decoupling your domain from your framework

Conclusion

- Consistent Interface
- **Self-explaining**
- Context Encapsulation
- Decoupling your domain from your framework

Conclusion

- Consistent Interface
- Self-explaining
- **Context Encapsulation**
- Decoupling your domain from your framework

Conclusion

- Consistent Interface
- Self-explaining
- Context Encapsulation
- **Decoupling your domain from your framework**

Thanks for Your Listening

@tonytonyjan

References

- https://8thlight.com/blog/uncle-bob/2012/08/13/the-clean-architecture.html
- https://www.wikiwand.com/en/Domain-driven_design
- https://github.com/cypriss/mutations
- https://github.com/collectiveidea/interactor
- https://github.com/tonytonyjan/interactor2

Q&A

Entity-Boundary-Interactor