

Rails config_for

March 9, 2015 Rails.MN

Tamara Temple <tamouse@gmail.com>

Typical Config File

```
default: &default
  user: nobody
  link: http://example.com
  environment: :sandbox

development:
  <<: &default
  datastore: badger.dev

test:
  <<: &default
  datastore: badger.test

production:
  user: <%= ENV[ 'BADGER_USER' ] %>
  link: <%= ENV[ 'BADGER_LINK' ] %>
  environment: :production
  datastore: <%= ENV[ 'BADGER_STORE' ] %>
```

A common idiom

- In an initializer:

```
require "yaml"  
require "erb"
```

```
MyCoolLib.configuration =  
  YAML.load(  
    ERB.new(  
      File.read(  
        Rails.root.join("config/my_cool_lib.yml")  
      ).result  
    )[Rails.env]
```

config_for

- In 4.2, they added **config_for**:

[http://api.rubyonrails.org/classes/Rails/
Application.html#method-i-config_for](http://api.rubyonrails.org/classes/Rails/Application.html#method-i-config_for)



A new idiom

```
MyCoolLib.configuration =  
  Rails.application.config_for(:my_cool_lib)
```

Validation Contexts

March 9, 2015 Rails.MN

Tamara Temple <tamouse@gmail.com>

What if you don't want to run a validation sometimes?

- Turn off validations: `save(validate: false)`
- Conditional validations:
`validates_presence_of :name, if:`
`Proc.new{|m| m.new_record?}`
- Contextual validations:
`validates_presence_of :name,`
`on: :create`

Make your own contexts

- In your model:
 - `validates_presence_of :address,`
`on: :final_submit`
- In a controller or service model:
 - `save(context: :final_submit)`

Deeper explanation

- Justin Weiss, prolific blogger and Rails enthusiast, wrote "A Lightweight Way to Handle Different Validation Contexts":
<http://www.justinweiss.com/blog/2014/09/15/a-lightweight-way-to-handle-different-validation-situations/>

