Cheatography

FactoryBot Cheat Sheet

by Ash Angell via cheatography.com/120958/cs/21962/

Defining factories

```
FactoryBot.define do
   # Default
   # It will use the User class
    factory :user do
        fir st_name { " Joh n" }
        las t_name { " Doe " }
        admin { false }
   end
   # Specifying the class
   # It will use the User class, instead of Admin
class
    factory :admin, class: User do
        fir st_name { " Adm in" }
        las t_name { " Use r" }
        admin { true }
   end
end
```

Options (transients)

```
factory :user do
    tra nsient do
        upcased true
   end
    after :create do |user, options|
        use r.n ame.up case! if option s.u pcased
   end
end

create (user, upcased: true)
```

Transient attributes will not get passed to the model, but will be available in after-create hooks.

Nested Factories

```
factory :user do
    fir st_name 'John'
    factory :sampl e_user do
        fir st_name { Faker: :Na me.f ir st_name }
   end
end
create :sampl e_user
```

Traits

```
FactoryBot.define do
    factory :post do
        title { 'An awesome post' }
       body { 'Lorem Ipsum...' }
        trait :published do
            status { :published }
       end
        trait :unpub lished do
            status { :draft }
       end

        trait :with_ com ments do
            aft er( :cr eate) do |post|
                cre ate _list :comment, 2,
todo_item: post
           end
       end
   end
end
# then in your test
let(:post) { create (:post, :publi shed) }
# or even with
let(:post) { create (:post, :publi shed, :with_ -
com ments) }
```

Trait helps you to remove duplication.

By **Ash Angell**
cheatography.com/ash-angell/

Published 7th March, 2020.
Last updated 8th March, 2020.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

FactoryBot Cheat Sheet

by Ash Angell via cheatography.com/120958/cs/21962/

Building factories

```
# Returns an User instance that's not saved
user = Factor yBo t.b uil d(: user)
# Returns a saved User instance
user = Factor yBo t.c rea te( :user)
# Returns a hash of attributes that can be used to
build an User instance
attrs = Factor yBo t.a ttr ibu tes _fo r(: user)
# Returns an object with all defined attributes
stubbed out
stub = Factor yBo t.b uil d_s tub bed (:user)
# Passing a block to any of the methods above will
yield the return object
Factor yBo t.c rea te( :user) do |user|
    use r.p ost s.c rea te( att rib ute s_f -
or( :post))
end
# Overriding attributes of a factory
user = Factor yBo t.b uil d(: user, first_ name:
" Joe ")
user.f irs t_name
# => " Joe "
# No matter which build strategy is used to
override attributes
user = Factor yBo t.c rea te( :user, first_ name:
" Joe ")
user.f irs t_name
# => " Joe "
```

Associations

```
# If the factory name is the same as the
association name, it's simple
factory :post do
    author
end
# You can also specify a different factory or
override attributes
factory :post do
   # ...
    ass oci ation :author, factory: :user,
last_name: " Wri tel y"
end
# Builds and saves a User and a Post
post = Factor yBo t.c rea te( :post)
```

Associations (cont)

```
> post.new_record? # => false
post.author.new_record? # => false
# Builds and saves a User, and then builds but does not save a Post
post = FactoryBot.build(:post)
post.new_record? # => true
post.author.new_record? # => false
```

Dependent attributes

```
factory :user do
    fir st_name { " Joe " }
    las t_name { " Blo w" }
    email { " #{f irs t_n ame }.# {la st_ nam e}@ -
exa mpl e.c om".d ow ncase }
end
```

Attributes can be based on the values of other attributes.

Aliases

```
factory :user, aliases: [:author, :commenter] do
    fir st_name { " Joh n" }
    las t_name { " Doe " }
    dat e_o f_birth { 18.yea rs.ago }
end
factory :post do
   # instead of associ ation :author, factory:
:user
    author
    title { "How to read a book effect ive ly" }
   body { " There are five steps involv ed."  }
end
factory :comment do
   # instead of associ ation :comme nter, factory:
:user
    com menter
   body { " Great articl e!" }
end
```

Aliases allow to use named associations more easily.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com