
Machinist 2

Fixtures aren't fun. Machinist is.

- Home page
- Google group, for support
- Bug tracker, for reporting Machinist bugs

If you want Machinist 1, go [here](#).

Introduction

Note: Machinist isn't under active development. See the Status section below for more info.

Machinist makes it easy to create objects for use in tests. It generates data for the attributes you don't care about, and constructs any necessary associated objects, leaving you to specify only the fields you care about in your test. For example:

```
1 describe Comment, "without_spam scope" do
2   it "doesn't include spam" do
3     # This will make a Comment, a Post, and a User (the author of the
4     # Post), generate values for all their attributes, and save them:
5     spam = Comment.make!(:spam => true)
6
7     Comment.without_spam.should_not include(spam)
8   end
9 end
```

You tell Machinist how to do this with blueprints:

```
1 require 'machinist/active_record'
2
3 User.blueprint do
4   username { "user#{sn}" } # Each user gets a unique serial number.
5 end
6
7 Post.blueprint do
8   author
9   title { "Post #{sn}" }
10  body { "Lorem ipsum..." }
11 end
12
13 Comment.blueprint do
14   post
15   email { "commenter#{sn}@example.com" }
16   body { "Lorem ipsum..." }
17 end
```

Installation

Upgrading from Machinist 1

See the wiki.

Rails 3

In your app's `Gemfile`, in the `group :test` section, add:

```
1 gem 'machinist', '>= 2.0.0.beta2'
```

Then run:

```
1 bundle
2 rails generate machinist:install
```

If you want Machinist to automatically add a blueprint to your blueprints file whenever you generate a model, add the following to your `config/application.rb` inside the Application class:

```
1 config.generators do |g|
2   g.fixture_replacement :machinist
3 end
```

Rails 2

See the wiki.

Usage

Blueprints

A blueprint describes how to generate an object. The blueprint takes care of providing attributes that your test doesn't care about, leaving you to focus on just the attributes that are important for the test.

A simple blueprint might look like this:

```
1 Post.blueprint do
2   title { "A Post" }
3   body  { "Lorem ipsum..." }
4 end
```

You can then construct a Post from this blueprint with:

```
1 Post.make!
```

When you call `make!`, Machinist calls `Post.new`, then runs through the attributes in your blueprint, calling the block for each attribute to generate a value. It then saves and reloads the Post. (It throws an exception if the Post can't be saved.)

You can override values defined in the blueprint by passing a hash to make:

```
1 Post.make!(:title => "A Specific Title")
```

If you want to generate an object without saving it to the database, replace `make!` with `make`.

Unique Attributes

For attributes that need to be unique, you can call the `sn` method from within the attribute block to get a unique serial number for the object.

```
1 User.blueprint do
2   username { "user-#{sn}" }
3 end
```

Associations

If your object needs associated objects, you can generate them like this:

```
1 Comment.blueprint do
2   post { Post.make }
3 end
```

Calling `Comment.make!` will construct a Comment and its associated Post, and save both.

Machinist is smart enough to look at the association and work out what sort of object it needs to create, so you can shorten the above blueprint to:

```
1 Comment.blueprint do
2   post
3 end
```

If you want to override the value for post when constructing the comment, you can do this:

```
1 post = Post.make(:title => "A particular title")
2 comment = Comment.make(:post => post)
```

For `has_many` and `has_and_belongs_to_many` associations, you can create multiple associated objects like this:

```
1 Post.blueprint do
2   comments(3) # Makes 3 comments.
3 end
```

Named Blueprints

Named blueprints let you define variations on an object. For example, suppose some of your Users are administrators:

```
1 User.blueprint do
2   name { "User #{sn}" }
3   email { "user-#{sn}@example.com" }
4 end
5
6 User.blueprint(:admin) do
7   name { "Admin User #{sn}" }
8   admin { true }
9 end
```

Calling:

```
1 User.make!(:admin)
```

will use the `:admin` blueprint.

Named blueprints call the default blueprint to set any attributes not specifically provided, so in this example the `email` attribute will still be generated even for an admin user.

You must define a default blueprint for any class that has a named blueprint, even if the default blueprint is empty.

Blueprints on Plain Old Ruby Objects

Machinist also works with plain old Ruby objects. Let's say you have a class like:

```
1 class Post
2   extend Machinist::Machinable
3
4   attr_accessor :title
5   attr_accessor :body
6 end
```

You can blueprint the Post class just like anything else:

```
1 Post.blueprint do
2   title { "A title!" }
3   body  { "A body!" }
4 end
```

And `Post.make` will construct a new `Post`.

Other Tricks

You can refer to already assigned attributes when constructing a new attribute:

```
1 Post.blueprint do
2   author { "Author #{sn}" }
3   body   { "Post by #{object.author}" }
4 end
```

More Details

Read the code! No, really. I wrote this code to be read.

Check out the specs, starting with the spec for `Machinable`.

Compatibility

I've tested this with:

Ruby versions: 1.8.7, 1.9.2, 1.9.3, 2.0.0 Rails versions: 2.3, 3.0, 3.2

It may well be happy with other versions too, but I'm not promising anything. Compatibility patches are welcome.

Developing

The Machinist specs and source code were written to be read, and I'm pretty happy with them. Don't be afraid to have a look under the hood!

If you want to submit a patch:

- Fork the project.
- Make your feature addition or bug fix.
- Add tests for it. This is important so I don't break it in a future version unintentionally.

-
- Commit, do not mess with rakefile, version, or history. (if you want to have your own version, that is fine but bump version in a commit by itself I can ignore when I pull)
 - Send me a pull request. Bonus points for topic branches.

Status

In active use in a number of large Rails 2 and 3 apps.

Development is sporadic at best, as I find myself with less and less need for factories in tests. See Bo Jeanes' excellent article on the topic.

If anybody wants to take over maintenance, let me know.

Contributors

Machinist is maintained by Pete Yandell (pete@notahat.com, @notahat)

Other contributors include:

Marcos Arias, Jack Dempsey, Jeremy Durham, Clinton Forbes, Perryn Fowler, Niels Ganser, Jeremy Grant, Jon Guymon, James Healy, Ben Hoskings, Evan David Light, Chris Lloyd, Adam Meehan, Kyle Neath, Lawrence Pit, Xavier Shay, T.J. Sheehy, Roland Swingler, Gareth Townsend, Matt Wastrodowski, Ian White

Thanks to Thoughtbot's Factory Girl. Machinist was written because I loved the idea behind Factory Girl, but I thought the philosophy wasn't quite right, and I hated the syntax.