



Leonard Garvey || @lgarvey

After the 15 minute blog...

@lgarvey - bottledup.net - reinteractive.net



How many people in the room are interested in teaching or mentoring new Rails/Ruby developers?

How many people would consider themselves as being new to Rails? Maybe you've recently attended InstallFest, RailsGirls or something more intensive like the General Assembly Ruby on Rails Course.



Learn / Teach

Ruby Koans
tryruby.org

guides.rubyonrails.org

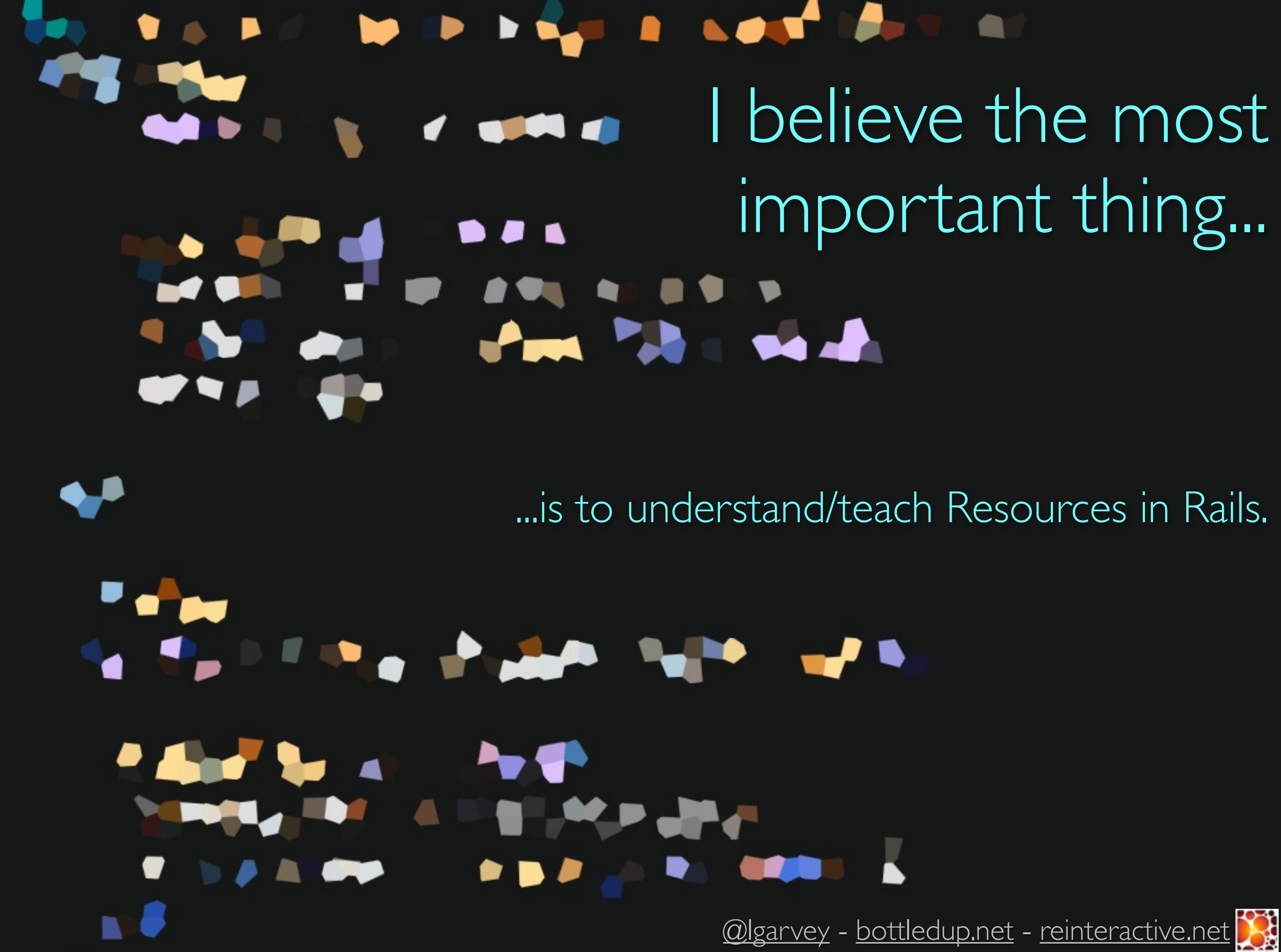
InstallFest/Development Hub

@lgarvey - bottledup.net - reinteractive.net



Programming is a profession for people who truly love to learn. The amount of things you “absolutely have to know” as a new developer is mind blowing. But assuming you’re just getting started and you’ve just finished your blog intro.

1. Learn Ruby. I really love the Ruby Koans, this is probably going to suit those people who have a programming background. It’s probably well worth going through the tutorial at <http://tryruby.org> too just to make sure you’ve got the basics down.
2. Learn some HTML, HTTP and CSS. Ultimately you want to build websites and Rails simply delivers HTML to a user.
3. Don’t be overwhelmed by the ALL THE THINGS! Start small and build your way up, everything you learn now will be useful later on. I spent years working in Perl and Bash, I don’t do so much programming in either anymore but I do still work with Linux which I worked in while doing Perl and Bash.



I believe the most
important thing...

...is to understand/teach Resources in Rails.

@lgarvey - bottledup.net - reinteractive.net



There are a few ideas that I'd like to go over here:

To all the new Rails developers you might be thinking “great I can build a blog, but what I actually want to build is Widget X!”. Through an understanding of Resources in Rails you'll come to appreciate that the main “thing” in a blogging application is the Post. The Post contains all the important data in the system, and most of the important behaviour in our blogging application concerns the Post. In every application there are going to be only a couple of really important Resources. Let's look at a couple of examples:

1. Twitter – The tweet is the main Resource. Everything revolves around Tweets.
2. Timesheet application – Entry. Everything revolves around the entry of your time. You could argue that the sheet itself is the main concern here. It's common for systems to have 2 main Resources with other important cross-cutting concerns (User is a common one).
3. A forum application (like Discourse) – Topic and Comments are going to be the most important Resources in this system.

My advice to everyone who is looking to build a Rails app is to focus on the main concern for your application. The absolute worst thing you can do is start off by building an authentication system (unless your system is an authentication provider?)

In the blog, the main Resource was the Post. The operations we perform on that Resource are CREATE, READ, UPDATE, DELETE. By encouraging this abstract thinking about your application it allows you to take the lessons from tutorials like the InstallFest one and apply it to the application you actually want to build.

Basically what I'm saying is that a blog application isn't so different from Twitter, a Timesheet app, or Widget XYZ SaaS app.

What is a resource?!

@lgarvey - bottledup.net - reinteractive.net



<http://stackoverflow.com/questions/4686945/what-is-a-resource-in-rails>

“Any object that you want users to be able to access via URI and perform [CRUD](#) (or some subset thereof) operations on can be thought of as a resource. In the Rails sense, it is *generally* a database table which is represented by a model, and acted on through a controller.”

Rails guides say:

“A resource is the term used for a collection of similar objects, such as posts, people or animals. You can create, read, update and destroy items for a resource and these operations are referred to as *CRUD* operations.”

I don't have good advice here except perhaps to say that finding your application metaphor will allow you to find your main application Resource. If anyone else has a better way of describing this then I sure would love to hear it. If the definitions provided by Stackoverflow or the Rails Guides make sense to you, that's great but I'd prefer something a little simpler for such a fundamental concept.

Once you understand the concept of a resource, you can easily trace how things flow through your Rails application from Router, to Controller, to Models to Views.



Build your thing

@lgarvey - bottledup.net - reinteractive.net





Build your thing

Build *any* thing

@lgarvey - bottledup.net - reinteractive.net



It's an old programmer joke that there are only 2 hard problems left in computer science.
Naming things, Cache invalidation and Off by 1 errors

I'd humbly submit that the absolute hardest problem is figuring out what to work on closely followed by having an idea but no clue how to turn that idea into a reality.

If you don't know what to work on try this:

1. Force yourself to complete every exercise the ruby.railstutorial.org. Discipline may be the most important thing here.
2. Do <http://projecteuler.net/>

You don't know how to get started. Here are some pointers:

1. Maybe ignore Rails for a bit and focus just on HTML. Build a very ugly mockup and don't worry about how it LOOKS.
2. If you need to do something really simple (maybe you want to collect some email addresses) consider using Sinatra instead of Rails. This is particularly relevant for people coming from, say, PHP.
3. Sketch a mockup on paper (or using a tool like Balsamiq). After you've done that try to identify what the main Resource is. If you can't find one then you're probably starting too complicated.
4. Take your sketch to someone you trust and ask them to help you identify the System metaphor. DevelopmentHub can be a great environment for this.
5. Maybe try to find a problem that can be solved without being a web application. There's plenty of value to be made just by automating that data parsing in your office and converting it into a CSV. Ruby is a good choice for this (it runs on Windows too).



Mentor

[@lgarvey](#) - [bottledup.net](#) - [reinteractive.net](#)



Volunteer at InstallFest, DevelopmentHub or RailsGirls.
Head along to the Hack night with the intention of just helping someone else out.

I've started pairing with people. As an aside if anyone wants to pair with me please come and chat with me afterwards.

Add yourself to the mentor list (although I'm unsure if this does anything?).

Just to be very clear if anyone has any questions about Rails or web development please feel free to ask me. My twitter handle is right there on the slide. I may not be able to answer instantly (or at all) but I should be able to point you in the right direction.



Asking for help

@lgarvey - bottledup.net - reinteractive.net



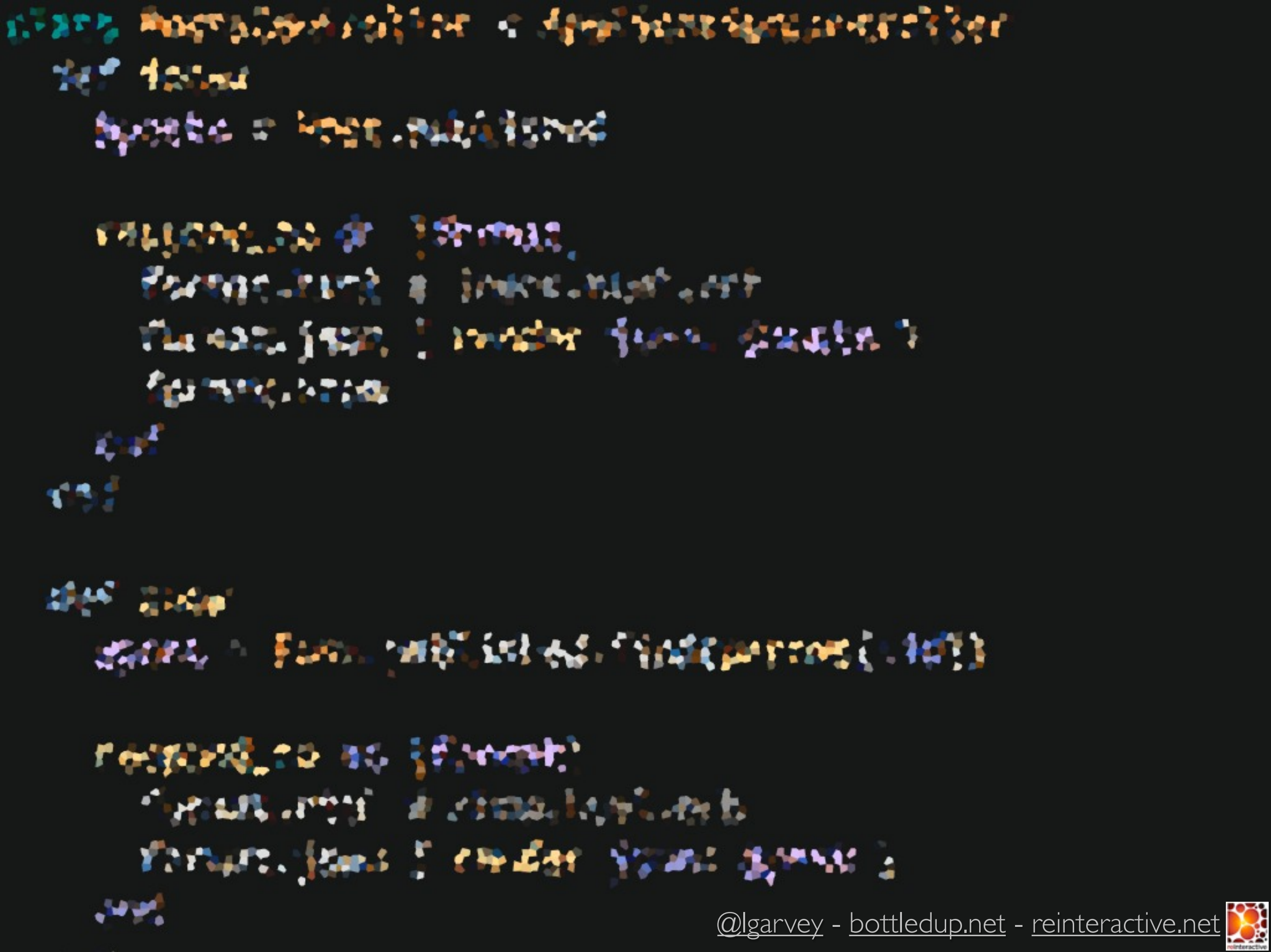
Knowing how to ask for help can be a vital step towards receiving the help you want, but as mentors we shouldn't be punishing people because they didn't say the magic words.

It's true it can be frustrating for a mentor to constantly be asked simple questions which could be solved very quickly with Google. In this case what I like to do is to respond: "Have you tried googling for X?"

For all you newbies out there the most important thing I'd like to tell you is that you should absolutely not be afraid to ask for help. There are plenty of lovely people who will gladly help you out.

If you're wanting to ask really good questions then what you should do is try to research your question as much as possible before asking it. Try to look at the error and put it up on <http://gist.github.com>. try to share as much code as you feel comfortable sharing so that your mentor can help you out as efficiently as possible. Try to be descriptive of what your problem is, and be prepared to respond to follow up questions.

There are several venues you can ask for help. You can try StackOverflow, IRC (freenode #rubyonrails), InstallFest, DevelopmentHub, My twitter, or almost anyone in this room.



@lgarvey - bottledup.net - reinteractive.net



To wrap up:

For those of you looking to learn more:

1. Get ready to do LOTS of learning. It honestly never stops.
2. If you're keen to do Rails (especially professionally) learn Ruby.
3. Don't be afraid to ask for help.
4. As a next step for learning Rails itself focus on understanding the concept of a Resource. Particularly focus on how to interact with a Resource in the request/response cycle.
5. If you want specific advice on what you should learn or focus on next, find a mentor (feel free to come ask me).

For those looking to teach:

1. Make yourself available. There are lots of different ways to do this. We're happy to have more mentors at InstallFest and DevelopmentHub.
2. One of the surprising things I've found from teaching is that almost everything we say as developers is some form of jargon. Be prepared to explain seemingly "simple" concepts that you take entirely for granted. This doesn't mean your student is stupid (quite the opposite in fact) but rather that over the years we've been doing this we've internalised a mental model.
3. You can easily start something of your own. Go for it!

```
class PostsController < ApplicationController
  def index
    @posts = Post.published

    respond_to do |format|
      format.html & index.html.erb
      format.json { render json: @posts }
      format.atom
    end
  end

  def show
    @post = Post.published.find(params[:id])

    respond_to do |format|
      format.html & show.html.erb
      format.json { render json: @post }
    end
  end
end
```

@lgarvey - bottledup.net - reinteractive.net



Hopefully as you progress all this complexity will begin to crystallize into something meaningful.


```
class PostsController < ApplicationController
  def index
    @posts = Post.published

    respond_to do |format|
      format.html # index.html.erb
      format.json { render json: @posts }
      format.atom
    end
  end
end
```

Questions? Comments?

```
def show
  @post = Post.published.find(params[:id])

  respond_to do |format|
    format.html # show.html.erb
    format.json { render json: @post }
  end
end
```

@lgarvey - bottledup.net - reinteractive.net



Thanks.

I've already posted these slides up on speakerdeck, I'll post a PDF with the speaker notes too to provide some better context.

Yes the whole background was one big setup.