[Overview](#h.6a3cyw5l2xux)

[Lesson 3.2: Gems & Scaffolds](#h.u09aq347qppk)

[Gem of the Day](#h.9ek0iyq1p7p2)

[Gems](#h.ti9o1rkmvvc3)

[What are Gems?](#h.hfcu7i69p4ky)

[Adding Gems to your Projects](#h.iufl3ujib3r9)

[Using Gems](#h.ap9xkmjbjxt6)

[Rails Generate Scaffold](#h.onvbjyxooeil)

[Buy Any One Column, Get Three More Free!](#h.iuceikezrswq)

[Migrations](#h.4lln9sxs1eo7)

[New Route(s)](#h.bpoxarcnvve4)

[Scaffolded Page Views](#h.8rahbu67u6uo)

[Index (/doggies)](#h.yoc1i0u8wlwe)

[Show](#h.kv36zlmfema9)

[New and Edit](#h.2vcekoyho6ht)

[Form](#h.krv01lbwmvq9)

[Activity: Adding an ID](#h.k60ga1gi05dd)

[Activity: Deleting a Record](#h.4dthkz89va67)

# Overview

* Lesson 3.2
  + Gems
  + Scaffolds

# Lesson 3.2: Gems & Scaffolds

## Gem of the Day

Quiet Assets <https://github.com/evrone/quiet_assets>

Font Awesome Rails <https://github.com/bokmann/font-awesome-rails>

## Gems

### What are Gems?

“Gems” are packages of Ruby code that extend or modify functionality in Ruby applications.

Commonly they’re used to distribute reusable functionality that is shared with other Rubyists for use in their applications and libraries.

Gems are what make Ruby on Rails so awesome compared to other languages and frameworks. That’s why a lot of what we talk about will be incorporating various gems into our projects.

### Adding Gems to your Projects

Generally, we never run gem install on the command line. Instead, we update our gemfiles and then run “bundle install”, or simply “bundle” for short.

Open your my\_awesome\_project project in Sublime, and navigate to the Gemfile. Let’s walk through what’s already there (do this). Now, let’s add some new gems:

# Don't want WEBrick

gem 'thin'

# Date and time formatting

gem 'stamp'

group :development do

# Quiet those assets down! Reduces noise in the logs

gem 'quiet\_assets'

end

Notice that we can tell Rails to only use certain gems in certain environments. We don’t want quiet assets in test or production because we want to have all data at our hands in those very important environments.

Now run bundle install, and re-start your server. It should look different since we are now using Thin. Also, if you navigate around your app, you will see the log on your command line will be much shorter.

It’s always a good idea to do a commit after updating gems and running bundle install. Now commit your changes.

### Using Gems

**Always check out the documentation** for gems to understand how to use them correctly. Frequently, this will be the ReadMe file in the GitHub repo, but it could also be the GitHub wiki, or RubyGems.org, or Ruby-Toolbox.

Let’s use stamp to make our Hello timestamp more human-readable.

Go to your Say Controller and edit the code in hello:

right\_now = Time.now

@current\_time = right\_now.stamp("12:00")

@current\_date = right\_now.stamp("February 1, 2015")

Now edit your views to use @current\_time and @current\_date. Voila! You just used a great gem. I would star it on Github. Now commit your changes.

## Rails Generate Scaffold

Now let’s do something new.

$ rails g scaffold Dog name:string breed:string age:integer

Remember when you created an Object 'class' in Ruby? Doesn't this seem awfully similar to that?!

Made a mistake? Don’t worry. Just delete your scaffold and try again:

$ rails destroy scaffold Dog

A Database is Born! Let’s break down that command:

* **rails g scaffold** - Scaffolding in Ruby on Rails refers to the auto generation of a simple set of a model, views and controller, usually for a single table.
* **Dog** - the name of our **data table**. Notice that it is Capitalized and Singular. This is very important. Rails knows how to pluralize, but it gets tripped up on singular words that end in “s” so try to get creative when you need to (“Focus” won’t work!)
* **name:string breed:string age:integer** - 'name', 'breed' and 'age' will be the **attributes** of your ‘Dog’; each is assigned a data type.

**Attributes** are like the column headings, or fields, of your table. Draw this:

*Dogs*

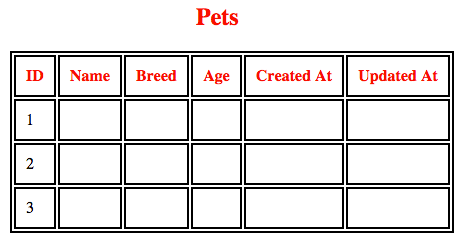
|  |  |  |
| --- | --- | --- |
| *name* | *breed* | *age* |
|  |  |  |
|  |  |  |

Each time you add a new dog, a new row in the table is created.

### Buy Any One Column, Get Three More Free!

Go to your database migrations folder. Who remembers how to get there?

Now take a look at your first migration. You will notice that Rails is a generous framework, and it gives three extra columns, just for stopping by...



* **id** - an integer from 1 thru ...
* **created\_at** - datetime when row was created
* **updated\_at** - datetime when the row was updated

### Migrations

Now, try to run your server and navigate to localhost:3000/dogs. Uh-oh, it doesn’t work. That’s because we never actually migrated our newly created table.

Migrations are a convenient way to alter your database schema over time in a consistent and easy way. You can think of **each migration as being a new 'version' of the database**.

A schema starts off with nothing in it, and each migration modifies it to add or remove tables, columns, or entries. Active Record knows how to update your schema along this timeline, bringing it from whatever point it is in the history to the latest version. Active Record will also update your db/schema.rb file to match the up-to-date structure of your database.

We will cover Migrations in more detail next week when we dive into Models and Databases. For now, just remember than you need to run **rake db:migrate** each time you do a scaffold or migration.

Run rake db:migrate and try navigating to localhost:3000/dogs again.

### New Route(s)

Rails also handles routing for us after we scaffold a resource. We see one simple line added to routes.rb.

resources :dogs  
  
That tiny bit of code actually maps to several pages. We'll explore what's going on behind the scenes here tomorrow.

Browse to localhost:3000/dogs (notice dogs is plural and lowercase).

Add data. Go ahead and add a few dogs using the form fields in your browser. Play around and get a feel for the various pages.

## Scaffolded Page Views

Let's take a look at the generated code on each page. (explain each and introduce erb which we will cover more)

### Index (/dogs)

Rails uses HTML tables to display the data on this page.

* Table headings hold the names of the 3 attributes we scaffolded (name, breed and age).
* And the table body loops through each pet and displays its attributes.

### Show

The show page consists of paragraphs to display the attributes of a specific dog.

### New and Edit

The new and edit pages are a little sparse. That's because the actual guts of the page are located on "form". The render tag is what displays the content on these views.

### Form

The "\_form" page is what's called a partial . A partial (denoted by an underscore) is reusable code that can be 'render'ed in many locations which keeps our code DRY. This code drops a form into our view and gathers data from the user.

### Activity: Adding an ID

Each pet id exists in our database but doesn't show by default in the view. Let's add an id column to our index.

<table>  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Name</th>  
 <th>Breed</th>  
 <th>Age</th>  
 <th colspan="3"></th>  
 </tr>  
 </thead>  
  
 <tbody>  
 <% @dogs.each do |pet| %>  
 <tr>  
 <td><%= pet.id %></td>  
 <td><%= pet.name %></td>  
 <td><%= pet.breed %></td>  
 <td><%= pet.age %></td>  
 <td><%= link\_to 'Show', pet %></td>  
 <td><%= link\_to 'Edit', edit\_dog\_path(dog) %></td>  
 <td><%= link\_to 'Destroy', dog, method: :delete, data: { confirm: 'Are you sure?' } %></td>  
 </tr>  
 <% end %>  
 </tbody>  
</table>

### Activity: Deleting a Record

Go ahead and delete the last pet record in your list. Then add a new pet.

Notice the id skips to the next number incrementally. Once an id is deleted the database does not reuse it.

## Migrations

Let’s add another column, or field, to our table using a migration. We want to add a column for the pet owners.

Run this command in your terminal, being very careful to use camel case:

rails g migration AddOwnerToDog owner:string

Remember, if you mess up, type “rails destroy” and then whatever you called the migration (exactly):

rails destroy migration AddOwnerToDog

This command tells Rails to add the field Owner to the table Dogs. Let’s open our migrations folder to double-check that it generated correctly. It should look like this:

class AddOwnerToDog < ActiveRecord::Migration

def change

add\_column :dogs, :owner, :string

end

end

Now, run rake db:migrate, then check your schema file to see how your database was updated. Then rails s, and check your views. Uh oh, we can’t see owners yet.

First, we need to make sure we call the owners field in the controller. Navigate to the bottom of your dogs controller and add :owner to the params list:

def dog\_params

params.require(:dog).permit(:name, :breed, :age, :owner)

end

Challenge: Add the owners column to your index view, show view, and form (hint: copy and then edit one of the other fields that also use a string).

Need more stuff!