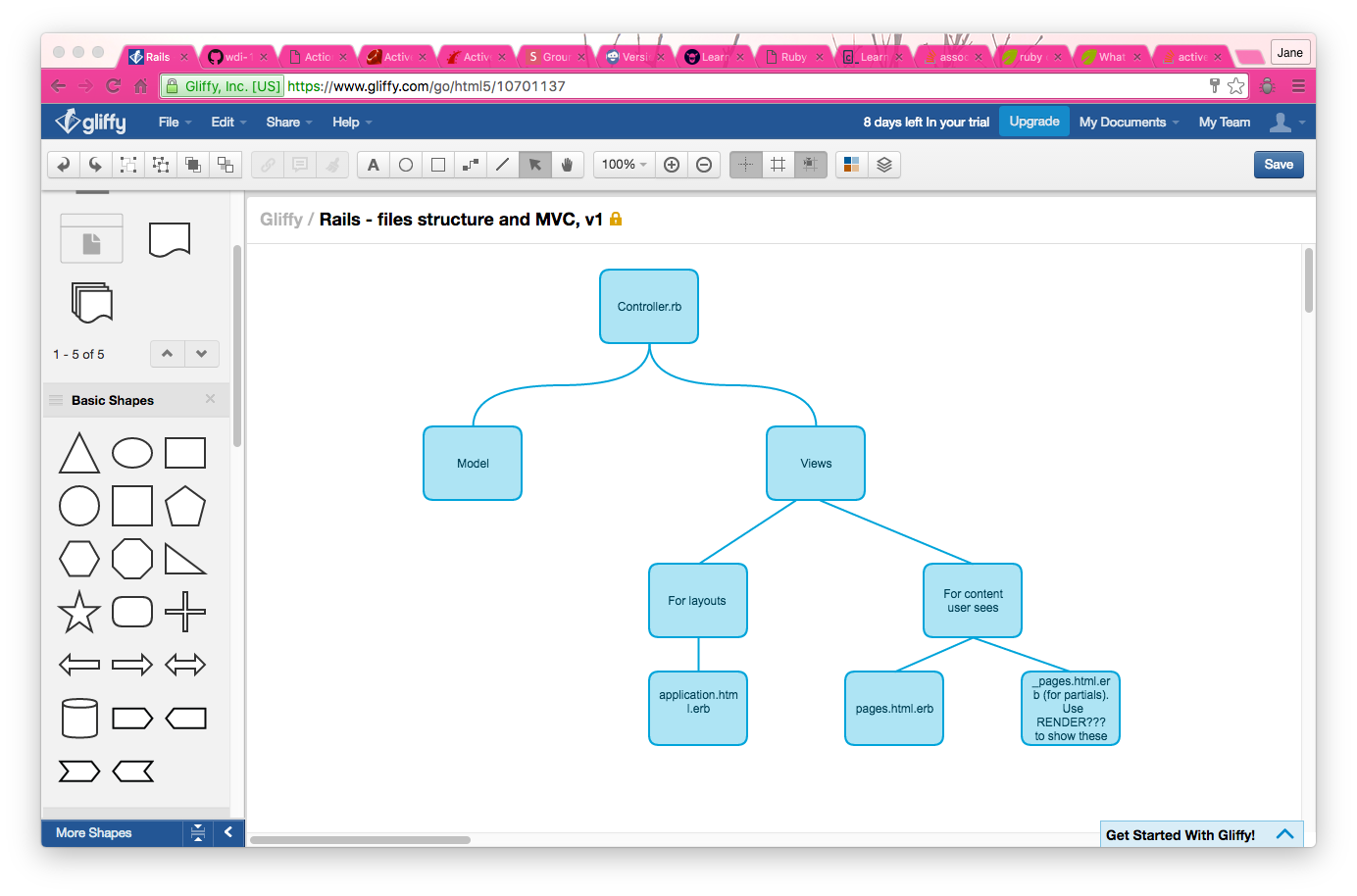
# Working in Rails – the steps

# Managing the views – infographic!

# Managing the models - infographic



# Questions

# High level steps

1. rails new app-name – need to add some thing else if I want to use postgres
2. cd app-name
3. Add gem 'pry-rails' and now annotate into your Gemfile (in the development group)
4. run bundle and then run annotate
5. run rake db:create
6. Generate your first migrations and models
   1. either rails g model Model name:string type:string etc. or:
   2. rails g migration create\_tables
7. Add the fields you need to that file - make sure you include t.timestamps
8. rails g model ModelNamebun
9. Generate your controllers and your views – either
   1. rails g controller Users index new create delete show etc. or:
   2. rails g controller Users
10. Add your methods into the controller
11. Create views that correspond with the method names in the view folder for that particular controller
12. Work out your routes file
13. Repeat this stuff as necessary

# Creating the rails app

| Stream | What | How |  |
| --- | --- | --- | --- |
|  | Create a new application | Console - rails new app\_namebb POSTGRESGemfile – add gemsConsole – bundle to include gems | rails new app\_nameif using postgresrails new BLAH –d postgresqlGemfile – add gemsConsole – bundle to include gems |
|  |  | Gemfile | Gem ‘pry-rails’Gem ‘annotate’ |
|  |  | Console – bundle to include gems | Bundle ORBundle install |
| DB/Model | Start the server | Console | CMD + D – to start a new version of the consolerails sORrails server |
| DB/Model – mostly uses rake commands ie rails make | Create the model for a table ie the schema | Console | QUESTION – is there a rake db:create hererails generate model Article title:string name:textThis will create a number of files including the article.rb and db/migrate/long number |
| DB/Model | Run the migration ie add the table model to your DB. | Console | rake db:migrate |
| DB/Model | Validation | .rb file | egvalidates : title, presence: true |
| DB/Model | Changing a column – need to generate a migration file (the one with the long name) and then rake it | console | If the migration name is of the form "AddXXXToYYY" or "RemoveXXXFromYYY" and is followed by a list of column names and types then a migration containing the appropriate add\_column and remove\_column statements will be created. |
|  |  |  | Can also change in the .rb migration file ? |
| DB Model | Join tables | console | rails g migration CreateJoinTableCustomerProduct customer project |
|  |  |  | create\_join\_table :producs, cagtegories |
|  | DB | seeds.rb | Add the stuff you want to add eg5.times do |i|  Product.create(name: "Product ##{i}", description: "A product.")endThen run rake db:seed |

# Managing the DB associations

## has\_many and belongs\_to and adding the sneaky table

Add associations to the models using the classes file.

The one that is the one ie belongs to the other, is the one that has the forieng key.

To add an extra column

Rails generate migration add\_artist\_id\_to\_works

Fill in migration

Rake db:migrate

|  |  |  |
| --- | --- | --- |
| Association | .rb schema |  |
| has\_many and belongs\_to | .rb  go in the models | Write the relevant associations in the files  has\_many :songs  belongs\_to :albums [Jane check that this is right]  DOES THIS NEED TO BE RAKED |
|  | add the foreign key to the relevant table  DO THIS FIRST | rails g migration add\_obligation\_id\_to\_clauses obligation\_id:integer  NB the use of the singular and plural |
| has\_and\_belongs\_to\_many | add to the .rb file of both models |  |
|  | create join table (sneaky table ) in alpha order | rails generate migration create\_mixtapes\_songs  make the id false |
| has\_many\_through | just goes in the relevant rb file | has\_many :genres => songs |

## Test associations in the console

Commands you can use once the association has been created

Rails c or rails consoler

W = Work.first

w.artist\_id = Artist.first.id

Artist.first.works

Works.first.artist

## Adding content into the join table

Use the append function on an instance of the join table ie

s1.artist << a1

s1.artist refers to s1 in the join artist table. << the standard Ruby append function

# Creating the controllers and the views

## Key principles in picture form

## Steps

* Generate your controllers and your views - either rails g controller Users index new create delete show etc. or:
  + rails g controller Users
  + Add your methods into the controller
  + Create views that correspond with the method names in the view folder for that particular controller
* Work out your routes file

## Key commands

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Views |  | in the config/routes.rb file | resources :model name in the singula this will create the various routes – have a look by rake routes |
|  |  |  |  |

### How do I create one form that collects data for multiple tables?

Need to put an instance variable in the relevant method in the controller, and then use as normal,