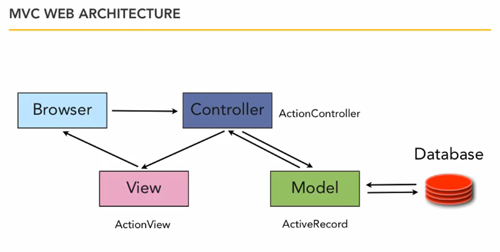
# GitHub Repository https://github.com/andrewprakash/WorkRepo.git

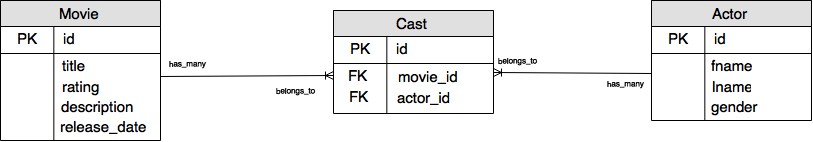
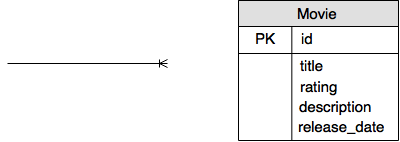
# Part 1 Database and Model Setup



Part 1

## Tasks

* Creating Database Migration
* Setup schema of tables
* Migrate database
* Setup Models

Database Schema

# After

# Before

## Migration Script

Navigate to the root of the project folder i.e **“/rottenpotatoes”**

To generate a migration script, by convention you should name the migration in plural “**actors**” and casts

**$ rails generate migration create\_actors**

**$ rails generate migration create\_casts**

Looking into the “**db/migrate**” folder you should see the migration ruby file created

The numbers in the front represent the date to the second

This allows for duplicate files to be also be present in case of a merge, because no two files will be created at the same time.

Opening the migration file you will see rails has already created a shell for adding tables to the database this is because you used **create\_** when you generated the migration file, therefore rails was smart enough to know what purpose the file is being generated for.

To create the tables you will need to add the required fields, the primary key is automatically generated when the tables is created

**“***create\_table\_actors.rb***”**

**class CreateActors < ActiveRecord::Migration**

**def up**

**create\_table ‘actors’ do |t|**

**t.string ‘fname’**

**t.string ‘lname’**

**t.string ‘gender’**

**end**

**def down**

**drop\_table ‘actors’**

**end**

**end**

**“***create\_table\_casts.rb***”**

**class CreateCasts < ActiveRecord::Migration**

**def up**

**create\_table ‘casts’ do |t|**

**t.integer ‘actor\_id’**

**t.integer ‘movie\_id’**

**end**

**def down**

**drop\_table ‘casts’**

**end**

**end**

## Insert tables into Database

To insert tables into the database

**$ rake db:migrate**

This will add the tables to the sqlite database.

Looking into the **db/schema.rb** file you will see the newly added script for the creating the tables.

## **1 .0 Detour**

At this point if you would like to confirm that the tables have been added to the database, use the tool from this link

[**http://sqlitebrowser.org/**](http://sqlitebrowser.org/)

To view the database open the “.sqlite” file from db folder using the tool.

## Models

At this point you now have to create the model for each of the tables you created i.e. actors and casts. Navigate to:

**/app/models**

The model’s file name convention is always singular **i.e. movie.rb,** therefore create the model file for each table

**actor.rb**

**cast.rb**

To define the class of each model the class name should always be the name of the file starting with uppercase. To access each of the attributes of the table you have to setup the attribute accessible method with the names of the columns.

**“***actor.rb***”**

**class Actor < ActiveRecord::Base**

**attr\_accessible :fname, :lname, :gender**

**end**

**“***cast.rb***”**

**class Cast < ActiveRecord::Base**

**attr\_accessible :movie\_id, :actor\_id**

**end**

## View Tables on Rails Console

In the terminal line navigate to project root

**$ rails console** or

**$ rails c**

You should now be in the console for rails where you can enter and test ruby code, the version of ruby being used is written to the left of your entry on the terminal

To initialise the tables use the following commands in the terminal

|  |  |
| --- | --- |
| **View all data in tables** | **View Schema of tables** |
| Actor.all | Actor |
| Cast.all | Cast |
| Movie.all | Movie |

## **1.1 Detour**

To add information to the tables in the console

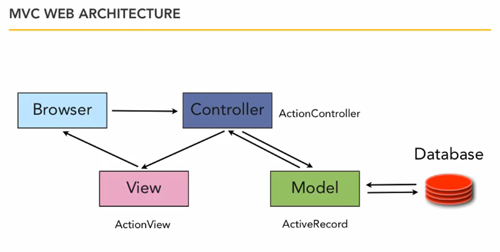
**$Actor.create(fname: ‘Jim’, lname: ‘Buchan’, gender:’Male’)**

This will commit the data to the database. To stage the data and not commit to database use

**$Actor.new()**

At this point controllers can be setup to interact with the model

# Part 2 Controller and View Setup



Part 2

## Tasks:

* Controllers Setup
* Routing Setup
* View Setup

It is recommended that you keep a separate terminal open for running your rails server and another for performing any working on the terminal.

It would also be helpful for you to run the application on your on browser after every code block entry done in this section, this will enable you to see how rails reacts to different connections between controller and model as well the tips it provides.

## Controllers

There needs to be a controller for each model that exists. Navigating to the controllers folder

**app/controllers**

Currently there is only two controller files in the folder, movies and application. Using the same naming convention the controllers for each model will be created.

Note the use of plural naming convention

**actors\_controller.rb**

**casts\_controller.rb**

## Routing

Each controller has its own view set based on **CRUD,** therefore routing for each view set needs to be setup.

Navigating into **app/config/** open the **routes.rb** file this file handles the routing of all views including the main root of the application.

Routing for each file could be entered individually, however rails provides an easier way to route all of the views of each view set.

Enter the following routes below **resources** **:movies**

**resources :actors**

**resources :casts**

## 2.0 Note:

This application does not use all the views (as will been shown later) that is defined when using “**resources**:”

It would be ideal to individually set out the routing that will be used.

[**http://guides.rubyonrails.org/routing.html**](http://guides.rubyonrails.org/routing.html)

Provides a guide on entering custom routing

If the routing is setup properly save file and enter into terminal

**$rake routes**

This will give a list of all the routes that can be used when navigating through the application



CRUD

## Views

For each controller/model there needs to be its allocated folder for CRUD.

**app/views**

This folder will house the all view files. Movies folder shows the basic view files for each controller.

To create the view files for actor and cast create the folders in the views folder

**views/actors**

**views/casts**

Since casts is the main interaction between movie and actors, it would be ideal to work from the cast view and controller before moving on to actors view and controller

Since it is not necessary to have all the views, hence not needing all the standard routing as described above.

Create the following files inside casts folder

**show.html.haml**

**edit.html.haml**

## Controllers Setup

After creating view files open the **“casts\_controller.rb”** file and define the controller class for casts with the method definitions

**class CastsController < ApplicationController**

**def show**

**end**

**def update**

**end**

**def edit**

**end**

**end**

This is all that will be needed for adding cast members and showing all the current cast members of a movie.

**def show** <= corresponds to what can be viewed on the **show.html.haml** view

**def edit** <= corresponds to what can be viewed on the **edit.html.haml** view

**def update** <= handles any submissions made from the **edit.html.haml** view

At this point there is no way to navigate to the cast of a movie, therefore we need to add a link for cast for each movie in the movies table.

Open **index.html.haml** from the movies view folder

Add another column header for Cast

**%th Cast**

Add the corresponding row data

**%td= link\_to “Cast”, cast\_path(movie)**

This generates a link to show all the casts members of the movie.

At this point if you run your rails server and click on the “cast” link on the movies table a blank table should appear, however if you look at the address bar you should see

**“localhost:3000/casts/movie\_id”**

Now we can setup the show method in the casts controller to show all the casts of a movie.

“*casts\_controller.rb*”

**class CastsController < ApplicationController**

**def show**

**@id = params[:id]**

**@cast = Cast.where(“movie\_id = ?”,@id)**

**@firstNameArray = Array.new**

**@lastNameArray = Array.new**

**@genderArray = Array.new**

**for i in @cast**

**@actor = Actor.find(i.actor\_id)**

**@firstNameArray.push(@actor.fname)**

**@lastNameArray.push(@actor.lname)**

**@genderArray.push(@actor.gender)**

**end**

**@movie = Movie.find(@id)**

**end**

**end**

**“@”** is used for globalisation which allows access to the variable on the corresponding view

The code block above gets the parameter id which is the movie id and searches through cast table for all actors that have same movie id. i.e. they are listed under the movie

## 2.1 Note:

To learn more the how to query database in rails

[**http://guides.rubyonrails.org/active\_record\_querying.html**](http://guides.rubyonrails.org/active_record_querying.html)

Shows examples of how to query the database in rails with corresponding SQL statements

We can display the information we have retrieved on the corresponding show view

*“casts/show.html.haml”*

**%ol.breadcrumb**

**%li**

**= link\_to 'All Movies ', movies\_path**

**%table#movies.table.table-hover**

**%thead**

**%tr**

**%th First Name**

**%th Last Name**

**%th Gender**

**%tbody**

**-@cast.each\_with\_index do |cast,index|**

**%tr**

**%td= @firstNameArray.at(index)**

**%td= @lastNameArray.at(index)**

**%td= @genderArray.at(index)**

**=link\_to “Add A Cast Member”,**

**edit\_cast\_path(@id),**

**:class=>”btn btn-success btn-large”**

As you can see the variables used in the controller are also being used here.

Since **@cast** object length is same as the arrays being used, we can loop through the object and access each element of the arrays based on the index of loop.

At this point if you run your application, clicking on the cast link from the movies page should take you to a page with an empty table and the “**Add A Cast Member**” button

Before we setup the edit view and method, we need to add some actors to our Actors tables.

Detour 1.1 shows you how to enter information into Actor table.

At this point the edit page can now be setup that will enable the ability to add actors to the cast members

*“casts\_controller.rb”*

21

1

**class CastsController < ApplicationController**

**def show**

**@id = params[:id]**

**@cast = Cast.where(“movie\_id = ?”,@id)**

**@firstNameArray = Array.new**

**@lastNameArray = Array.new**

**@genderArray = Array.new**

**for i in @cast**

**@actor = Actor.find(i.actor\_id)**

**@firstNameArray.push(@actor.fname)**

**@lastNameArray.push(@actor.lname)**

**@genderArray.push(@actor.gender)**

**end**

**@movie = Movie.find(@id)**

**end**

**def edit**

**@id = params[:id]**

**@cast = Cast.where(“movie\_id = ?”,@id)**

**@firstNameArray = Array.new**

**@lastNameArray = Array.new**

**@genderArray = Array.new**

**for i in @cast**

**@actor = Actor.find(i.actor\_id)**

**@firstNameArray.push(@actor.fname)**

**@lastNameArray.push(@actor.lname)**

**@genderArray.push(@actor.gender)**

**end**

**@movie = Movie.find(@id)**

**@actors = Actor.all**

**@actorsArray = Array.new**

**@actorsId = Array.new**

**for j in @actors**

**fullName = j.fname + “ “ + j.lname**

**@actorsArray.push(fullName)**

**@actorsId.push(j.id)**

**end**

**end**

**def update**

**getParam = params[:actorName]**

**getParam = getParam[:fname]**

**splitName = getParam.spilt**

**@actor = Actor.where(“fname = ? AND lname = ?”,splitName[0],splitName[1])**

**@cast = Cast.create!(movie\_id: params[:id],actor\_id: @actor[0][:id])**

**redirect\_to cast\_path(params[:id])**

**end**

**end**

1. Here the all the actors in database are extracted and concatenated, placed in an array which is then displayed as select drop down list on the edit view
2. Depending on the selection from the drop-down list the actor id is added to the cast list under the movie cast being edited

*“casts/edit.html.haml”*

**%ol.breadcrumb**

**%li**

**= link\_to 'All Movies ', movies\_path**

**%table#movies.table.table-hover**

**%thead**

**%tr**

**%th First Name**

**%th Last Name**

**%th Gender**

**%tbody**

**-@cast.each\_with\_index do |cast,index|**

**%tr**

**%td= @firstNameArray.at(index)**

**%td= @lastNameArray.at(index)**

**%td= @genderArray.at(index)**

**= form\_tag cast\_path(@id), :method => :put do**

**= select :actorName, :fname, @actorsArray**

**= submit\_tag, ‘Add Actor’, :class => ‘btn btn-success btn-large’**

**= link\_to ‘Cancel’, cast\_path(@id), :class => “ btn btn-default btn-large**

321

1. When the user selects the actor and clicks to add it to the cast list, it is sent to the update section of the cast controller, where it is inserted into the database. When redirected back to the show cast page the new cast member is shown

## Testing New Setup Locally

At this point you should be able go to the show section of each cast as well go to the edit section.

If you did detour 1.1 you should have the list actors in drop down list.

## Suggestions

HashMaps instead of Arrays

## Tasks Remaining

Ability to add actors to database

Edit actor data

Delete actor

View Actor Profile from cast table

Delete cast member from movie