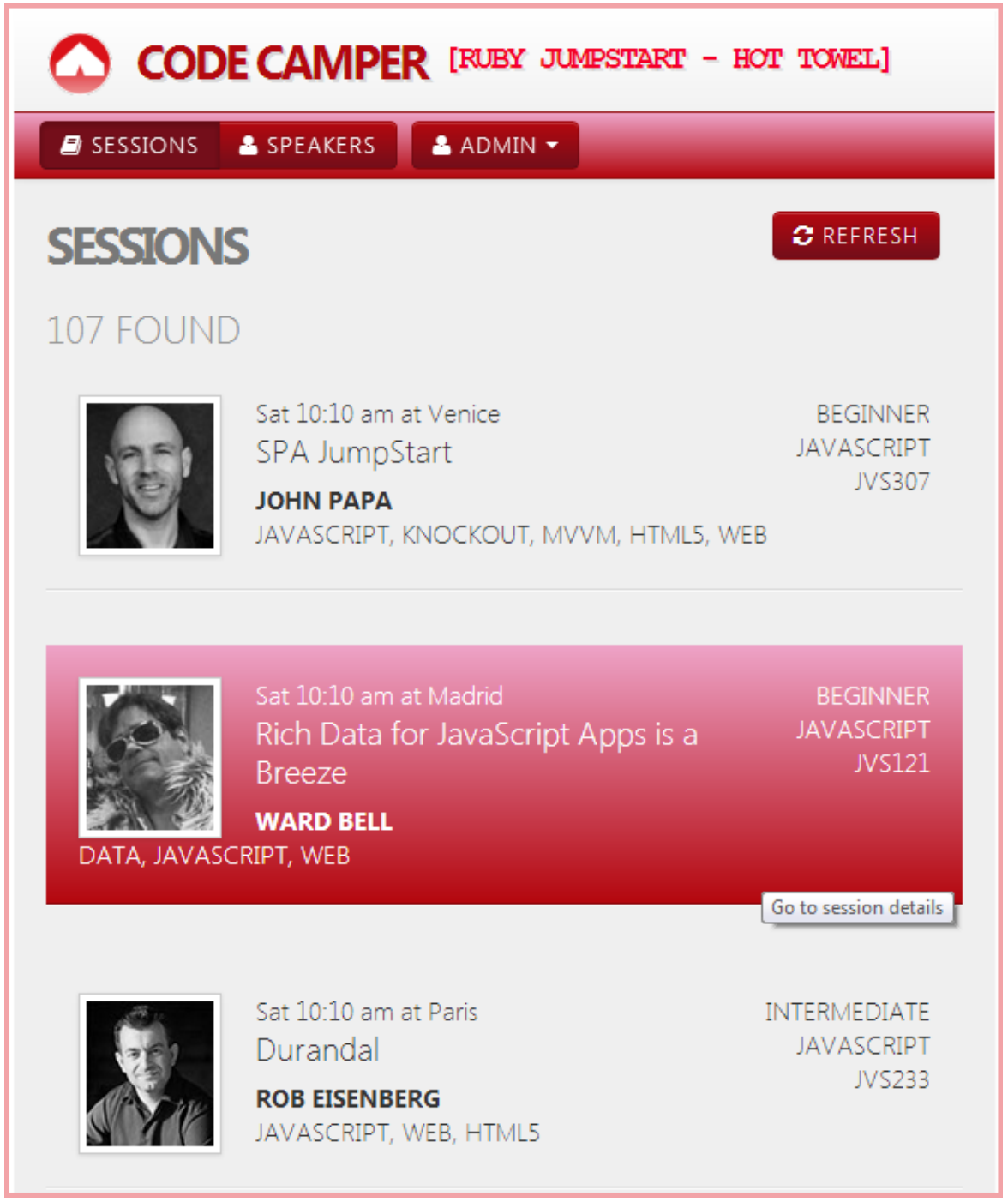
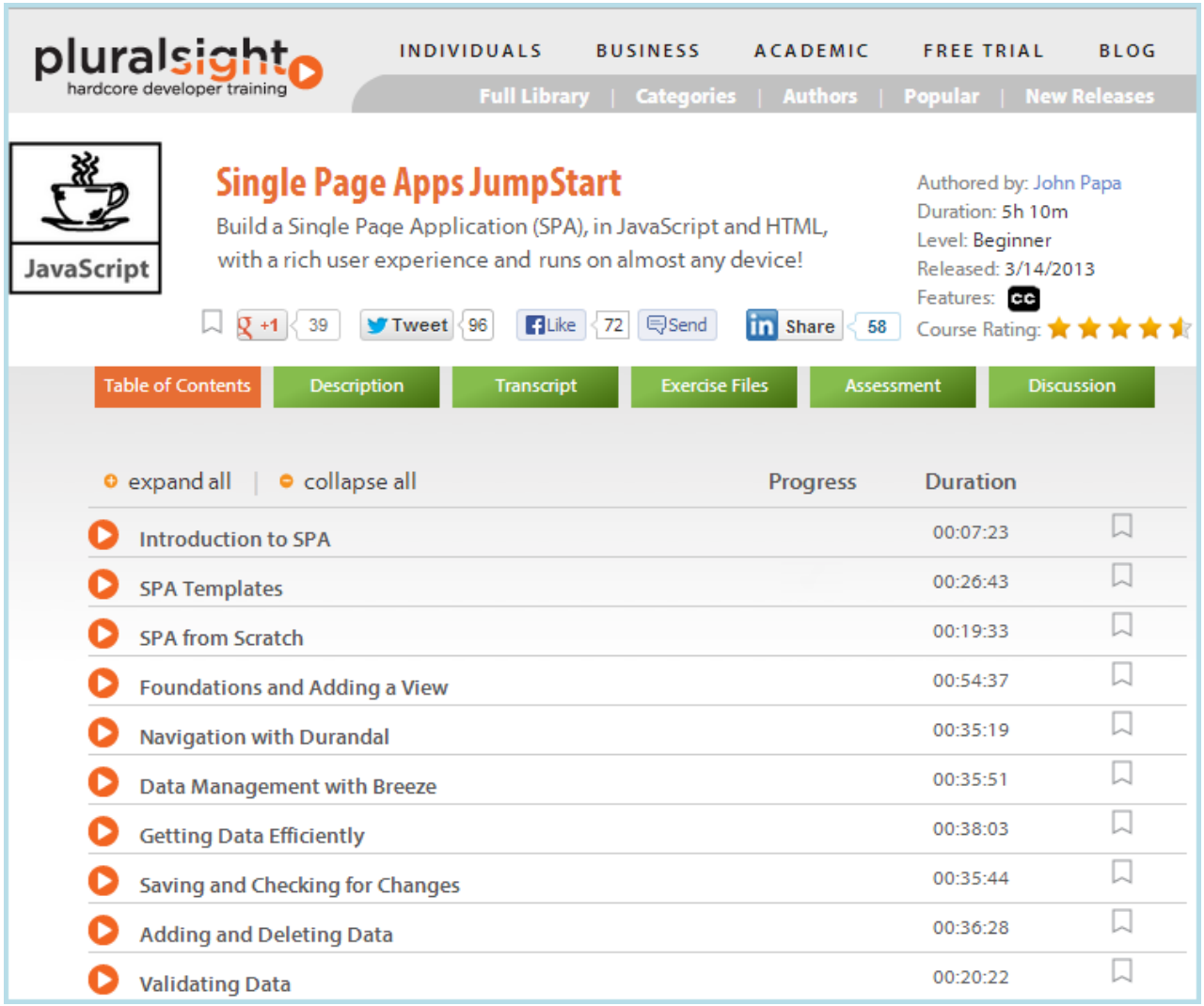


Single Page Apps with Breeze and Ruby

John Papa's Code Camper Jumpstart



//pluralsight.com/training/Courses/TableOfContents/single-page-apps-jumpstart https://github.com/johnpapa/PluralsightSpaJumpStartFinal

Features

- Multi-entity model
- Navigation properties / cached lookup lists
- Projections / partial entities
- Change notification
- Dirty checking
- Validation
- Save
- Local storage

Keys

- Client app is server agnostic (almost)
- Ruby on Rails (RoR) server
- Straight rails; no breeze.ruby components
- “REST” API
- Python web server
- No MS

Not anti-Microsoft

If it works in Ruby
it can work for you

~\ccjs_ruby\rails>bundle exec rails s

Run it

1. [Download from GitHub]
2. Setup install gems and create sample MySQL database
3. Start rails server `bundle exec rails s`

```
C:\Users\Ward\Documents\GitHub\Breeze\Samples\CCJS-Ruby\rails>bundle exec rails s
=> Booting WEBrick
=> Rails 3.2.13 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2013-09-11 14:27:22] INFO  WEBrick 1.3.1
[2013-09-11 14:27:22] INFO  ruby 1.9.3 (2013-06-27) [i386-mingw32]
[2013-09-11 14:27:22] INFO  WEBrick::HTTPServer#start: pid=31784 port=3000
```

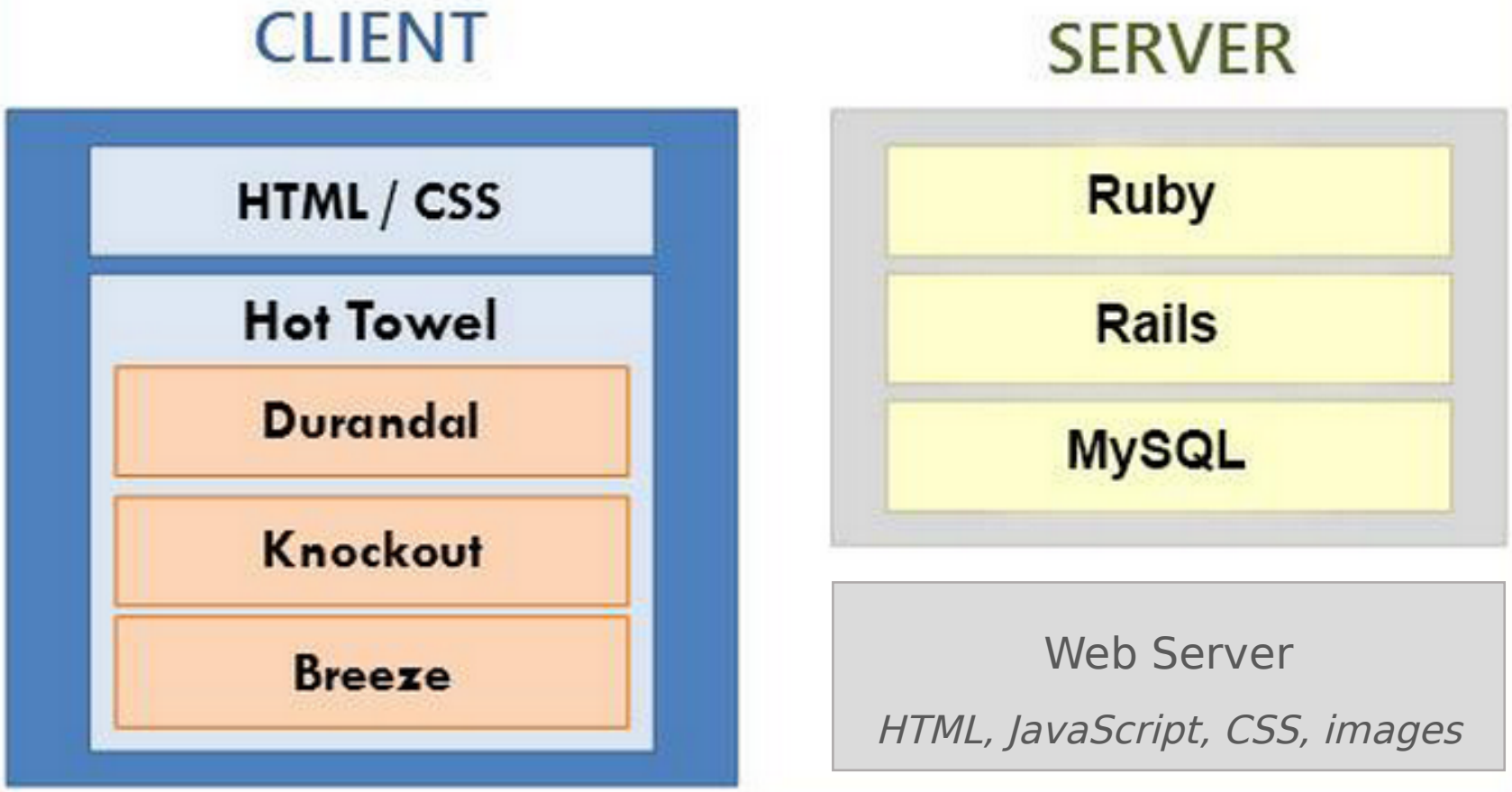
4. Start client application server `python -m http.server`

```
C:\Users\Ward\Documents\GitHub\Breeze\Samples\CCJS-Ruby\client>python -m http.server
Serving HTTP on 0.0.0.0 port 8000 ...
```

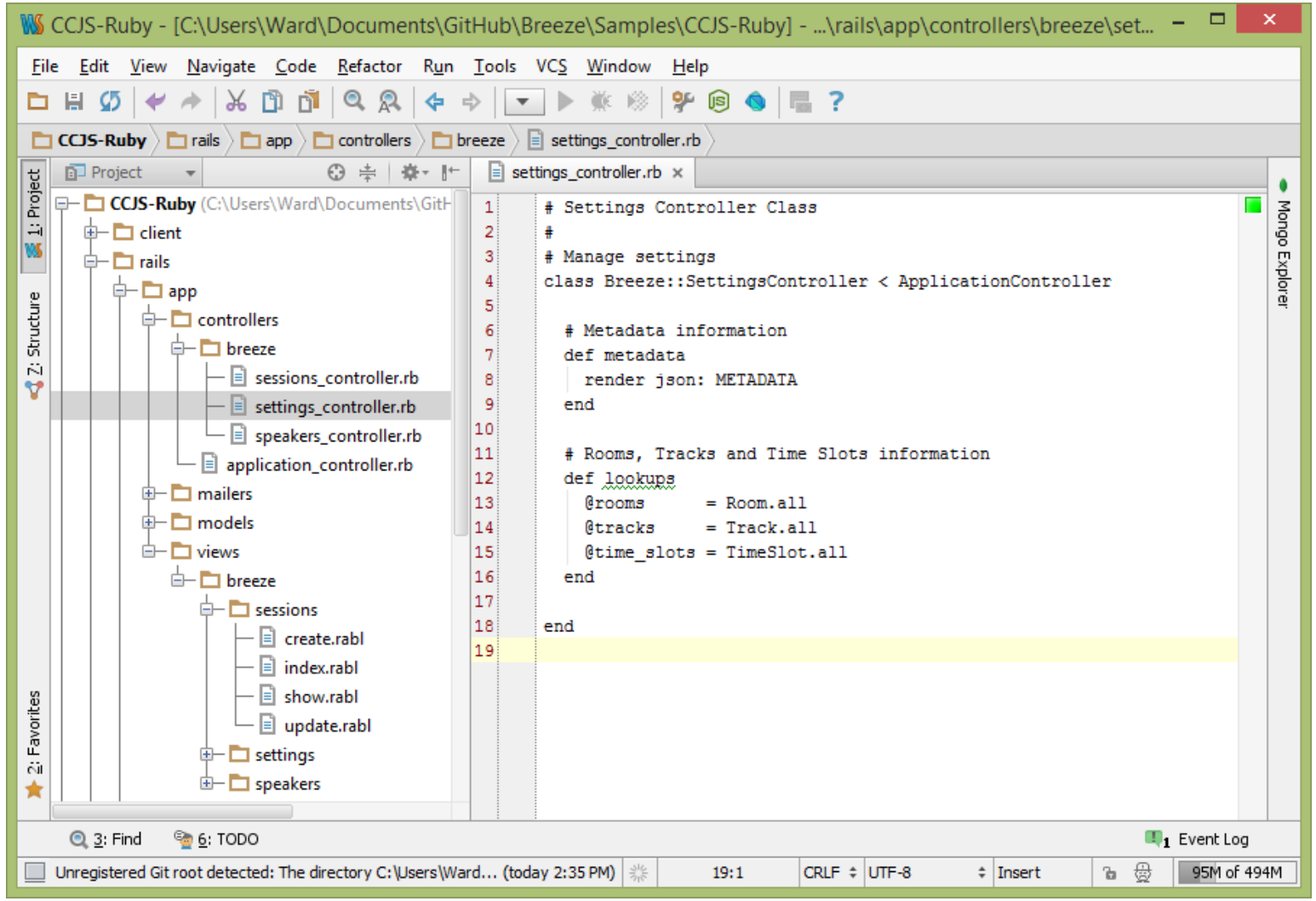
5. [illegible] `t:8000`

Demo

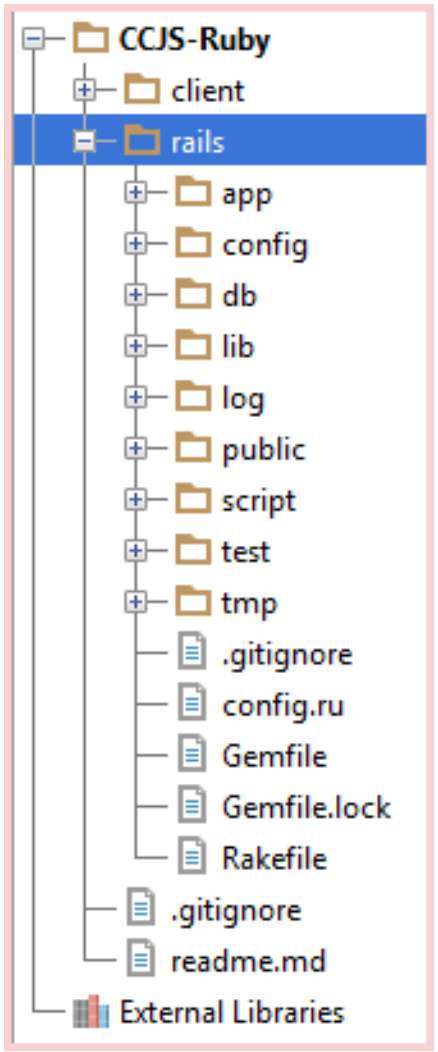
Architecture



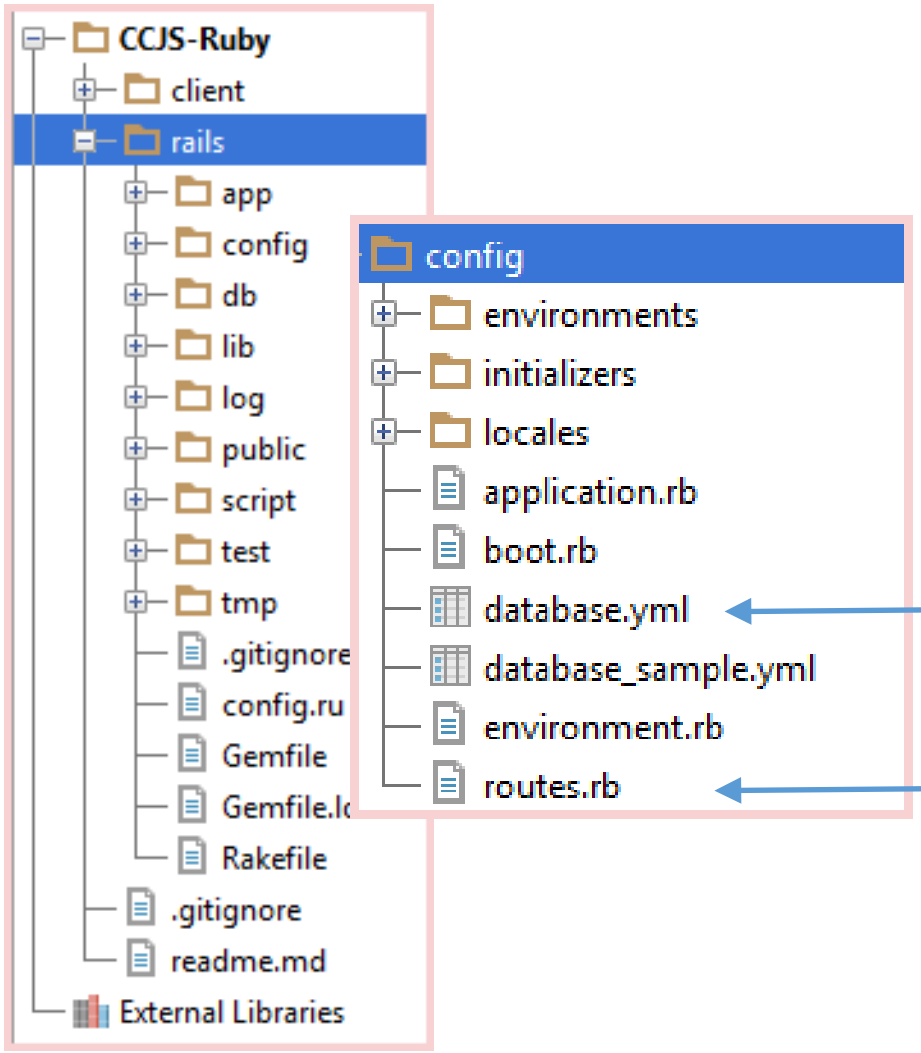
Code Tour


CCJS-Ruby - [C:\Users\Ward\Documents\GitHub\Breeze\Samples\CCJS-Ruby] - ...\rails\app\controllers\breeze\set...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
CCJS-Ruby rails app controllers breeze settings_controller.rb
Project
CCJS-Ruby (C:\Users\Ward\Documents\Gith
client
rails
app
controllers
breeze
sessions_controller.rb
settings_controller.rb
speakers_controller.rb
application_controller.rb
mailers
models
views
breeze
sessions
create.rabl
index.rabl
show.rabl
update.rabl
settings
speakers
settings_controller.rb
Settings Controller Class
#
Manage settings
class Breeze::SettingsController < ApplicationController
Metadata information
def metadata
render json: METADATA
end
Rooms, Tracks and Time Slots information
def lookups
@rooms = Room.all
@tracks = Track.all
@time_slots = TimeSlot.all
end
end
1: Project
2: Favorites
7: Structure
Mongo Explorer
3: Find
6: TODO
Event Log
Unregistered Git root detected: The directory C:\Users\Ward... (today 2:35 PM)
19:1
CRLF
UTF-8
Insert
95M of 494M

RoR Server


CCJS-Ruby
client
rails
app
config
db
lib
log
public
script
test
tmp
.gitignore
config.ru
Gemfile
Gemfile.lock
Rakefile
.gitignore
readme.md
External Libraries

RoR Server – Configuration

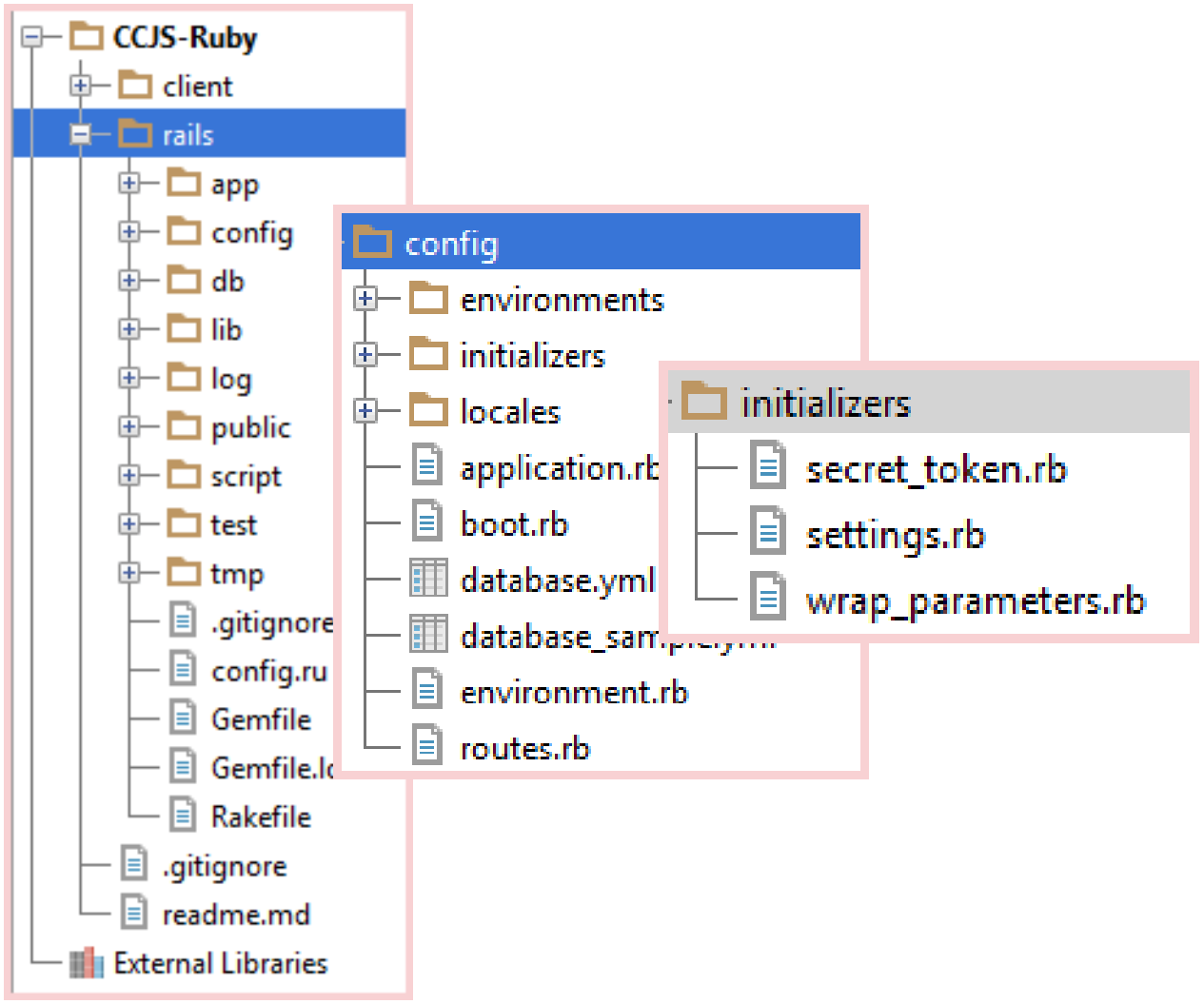


Locate the My SQL database

routes.rb

```
RubyCCJS::Application.routes.draw do
  get '/breeze/Metadata', to: 'breeze/settings#metadata'
  get '/breeze/Lookups',  to: 'breeze/settings#lookups'
  namespace :breeze do
    resources :Sessions, controller: 'sessions'
    resources :Speakers, controller: 'speakers'
  end
end
```

RoR Server – Configura

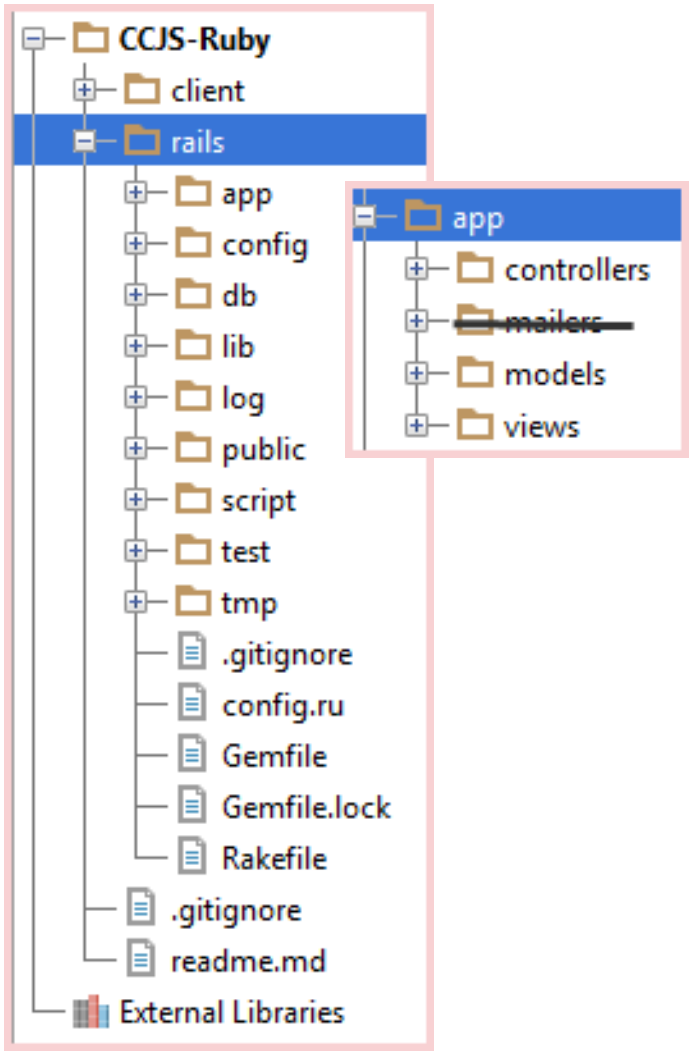


Breeze

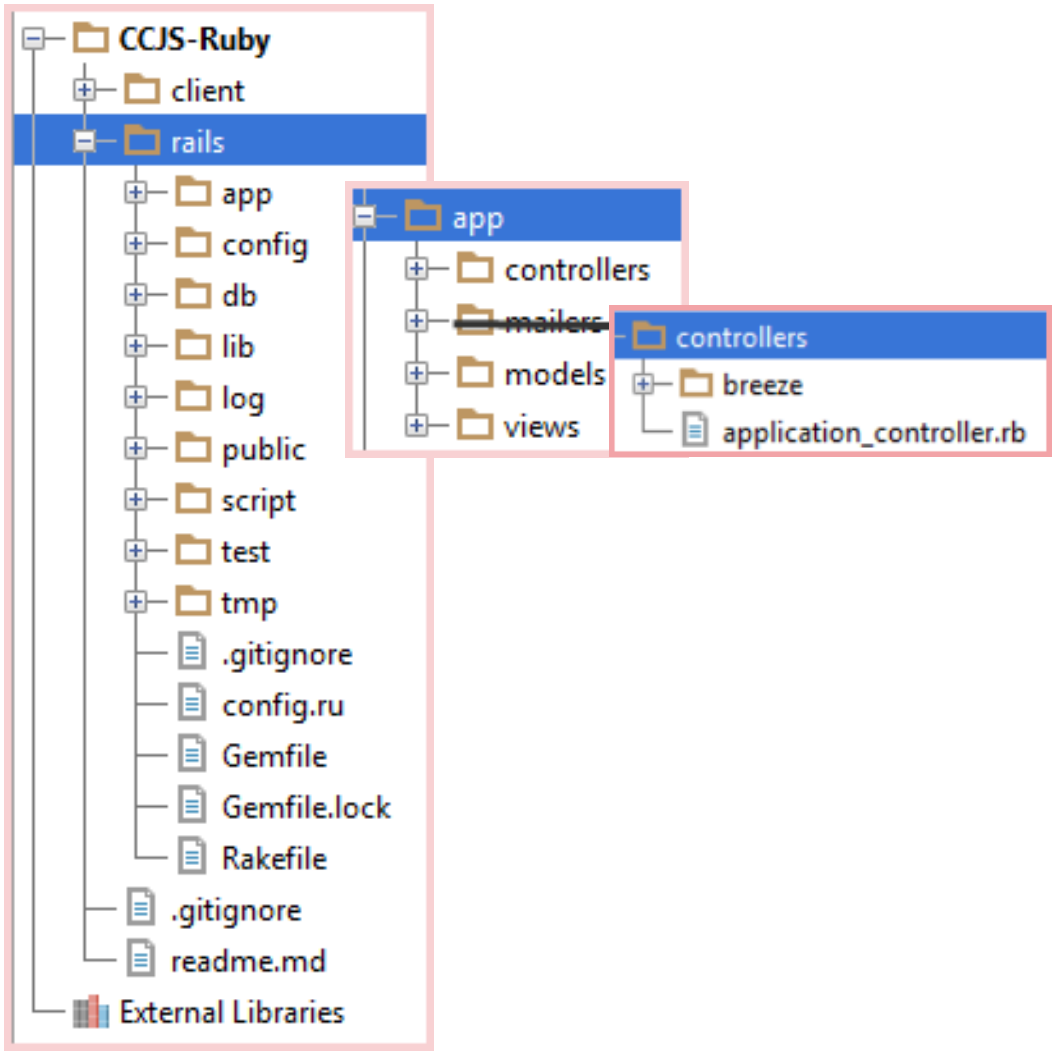
settings.rb

```
METADATA = '{
  "metadataVersion": "1.0.5",
  "namingConvention": "rubyNamingConvention",
  "localQueryComparisonOptions": "caseInsensitiveSQL",
  "dataServices": [
    {
      "serviceName": "breeze\/Breeze\/",
      "hasServerMetadata": true,
      "jsonResultsAdapter": "webApi_default",
      "useJsonp": false
    }
  ],
  "structuralTypes": [
    {
      "shortName": "Session",
      "namespace": "CodeCamper",
      "autoGeneratedKeyType": "Identity",
      "defaultResourceName": "Sessions",
      "dataProperties": [
        {
          "name": "id",
          "dataType": "Int32",
          "isNullable": false,
          "defaultValue": 0,
          "isPartOfKey": true,
          "validators": [
            {
              "name": "required"
            },
            {
              "name": "int32"
            }
          ]
```

RoR Server – Model View Controller



RoR Server – Controllers



application_controller.rb

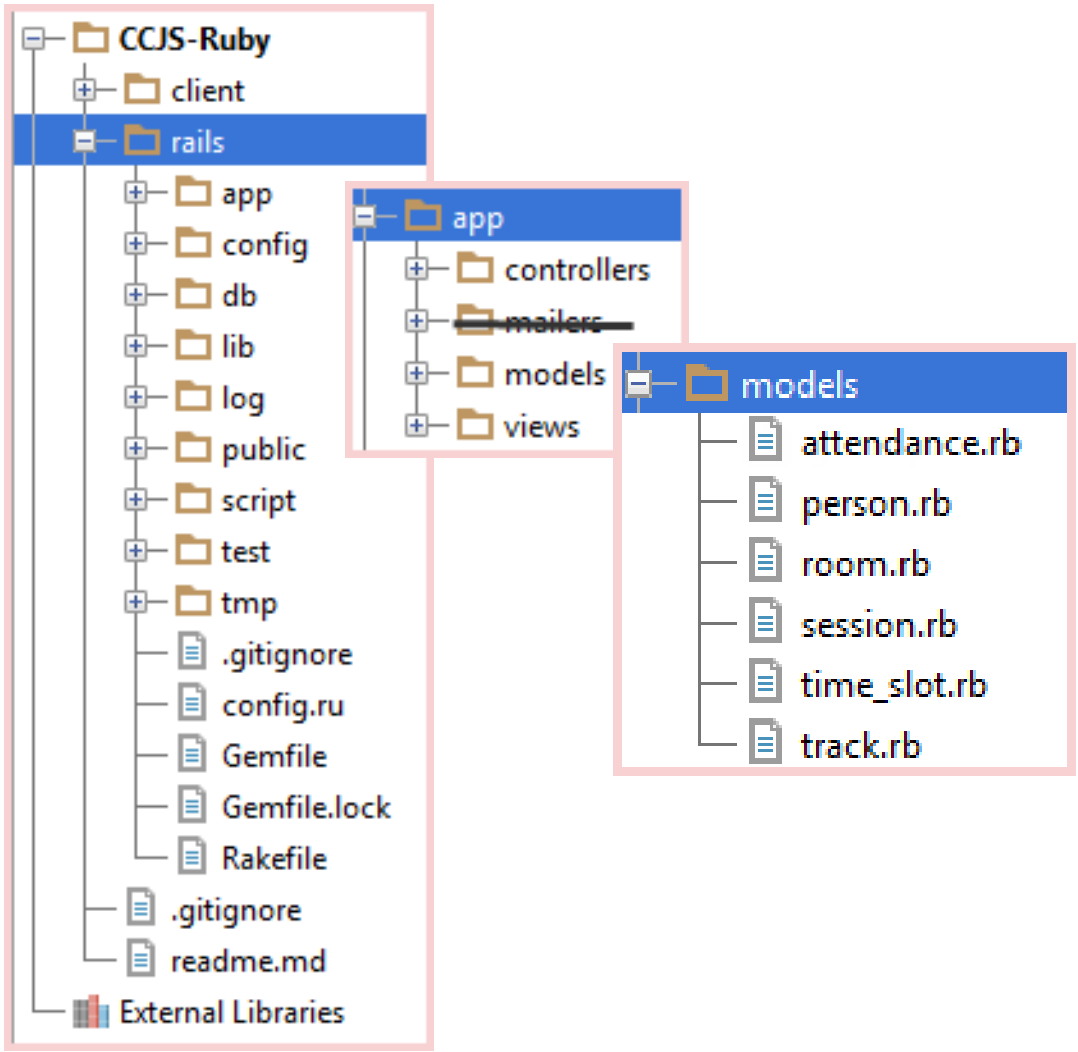
Cross Origin Resource Sharing

```
16      use Rack::Cors do
17        allow do
18          origins 'localhost:3000', '127.0.0.1:3000',
19                  /http:\/\/192\.168\.0\.\d{1,3}(:\d+)?/
20                  # regular expressions can be used here
21
22          resource '/file/list_all/', :headers => 'x-domain-token'
23          resource '/file/at/*',
24              :methods => [:get, :post, :put, :delete, :options],
25              :headers => 'x-domain-token',
26              :expose  => ['Some-Custom-Response-Header'],
27              :max_age => 600
28              # headers to expose
29          end
30
31          allow do
32            origins '*'
33            resource '/public/*', :headers => :any, :methods => :get
34          end
35        end
```

application.rb

```
62      # Enable CORS
63      config.middleware.use Rack::Cors do
64        allow do
65          origins '*'
66          resource '*', :headers => :any, :methods => [:get, :post, :put, :delete, :options]
67        end
68      end
69    end
70  end
```

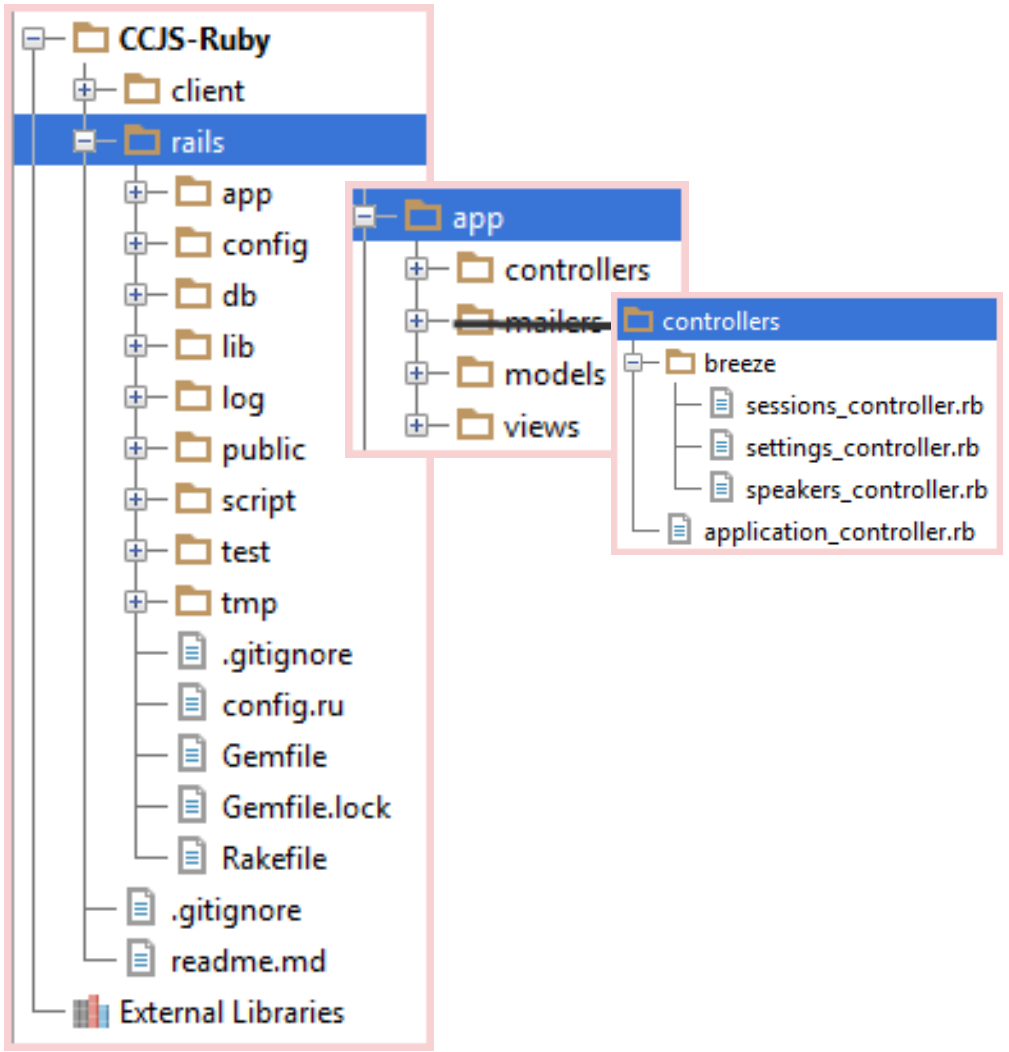
RoR Server – Models



session.rb

```ruby
# Session Class
#
# Session active record model
class Session < ActiveRecord::Base

  # Setup accessible attributes
  attr_accessible :title, :code, :speaker_id, :track_id, :time_slot_id,
  :room_id, :level, :tags, :description

  # Associations
  belongs_to :speaker, foreign_key: :speaker_id, class_name: 'Person'
  belongs_to :room
  belongs_to :time_slot
  belongs_to :track
  has_many   :attendances
end
```

RoR Server - Contr



Sessions Controller

sessions_controller.rb

```ruby
# Sessions Controller Class
#
class Breeze::SessionsController < ApplicationController

  # Sessions list      GET all
  def index
    order  = get_order('time_slot_id')
    @attributes = get_selected_attributes('Session')
    @type = params['$select'] ?
        'CodeCamper.SessionPartials, CCJS.Model' :
        'CodeCamper.Sessions, CCJS.Model'
    @sessions = Session
                    .joins('INNER JOIN people AS speakers ON sessions.speaker_id = speakers.id')
                    .select('DISTINCT sessions.*')
                    .order(order)
  end

  # Sessions details   GET by id
  def show
    @session = Session.find(params[:id])
  end

  # Create session     PUT
  def create
    @session = Session.create(params[:session])
  end

  # Update session     POST
  def update
    @session = Session.update(params[:id], params[:session])
  end

  # Delete session     DELETE
  def destroy
    Session.destroy(params[:id])
    render(json: {}, :nothing => true, :status => :no_content)
  end

end
```

Sessions Controller – Index (get all)

Projection (selected fields)

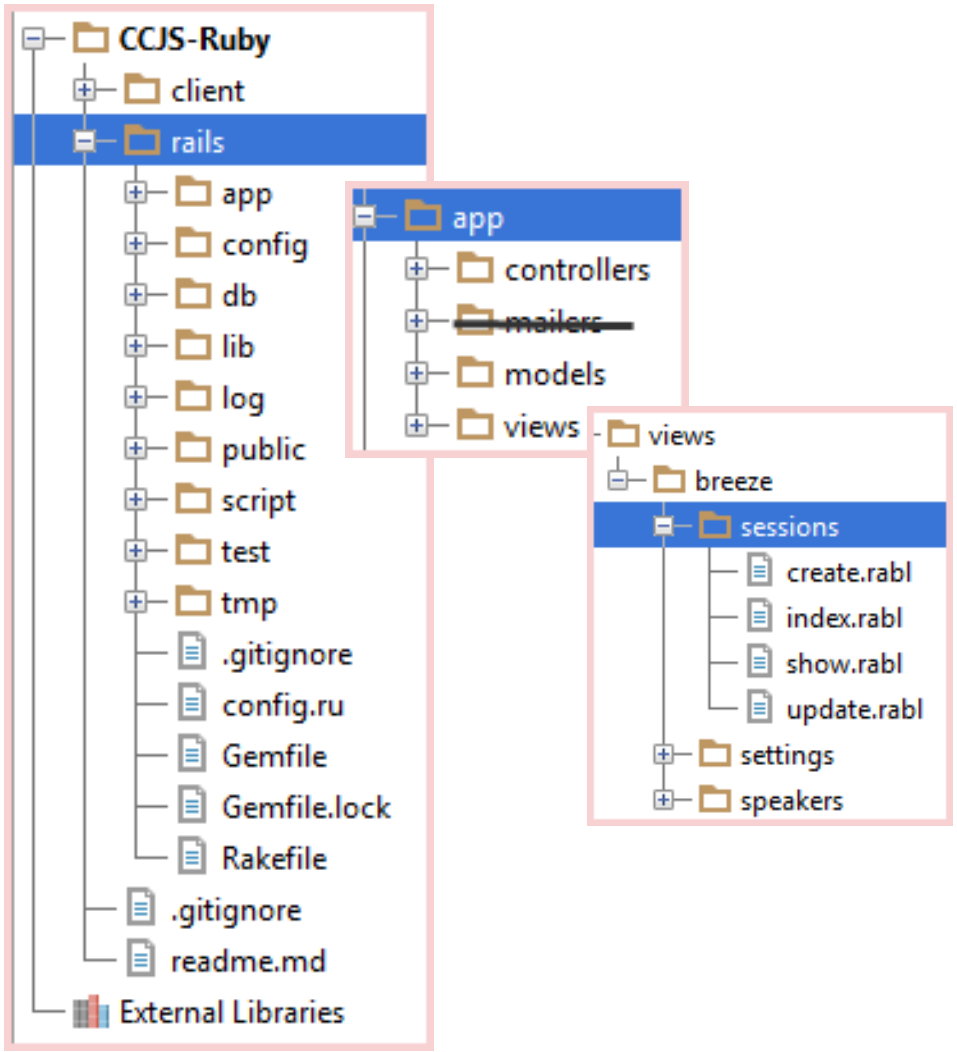
Partial entity if $select in query

```
# Sessions list
def index
  order  = get_order('time_slot_id')
  @attributes = get_selected_attributes('Session')
  @type = params['$select'] ?
      'CodeCamper.SessionPartials, CCJS.Model' :
      'CodeCamper.Sessions, CCJS.Model'
  @sessions = Session
                  .joins('INNER JOIN people AS speakers ON sessions.speaker_id = speakers.id')
                  .select('DISTINCT sessions.*')
                  .order(order)
end
```

Default sort order is time_slot_id but client typically wants by 'timeslot, track, speaker name'

Make speaker available in case sorting on speaker name

RoR Server – Session Views



create.rabl

```
1  object @session
2  attributes(*@session.attributes.keys)
```

index.rabl

```
1  collection(@sessions, object_root: false)
2  node('$type') { @type }
3  attributes(*@attributes)
```

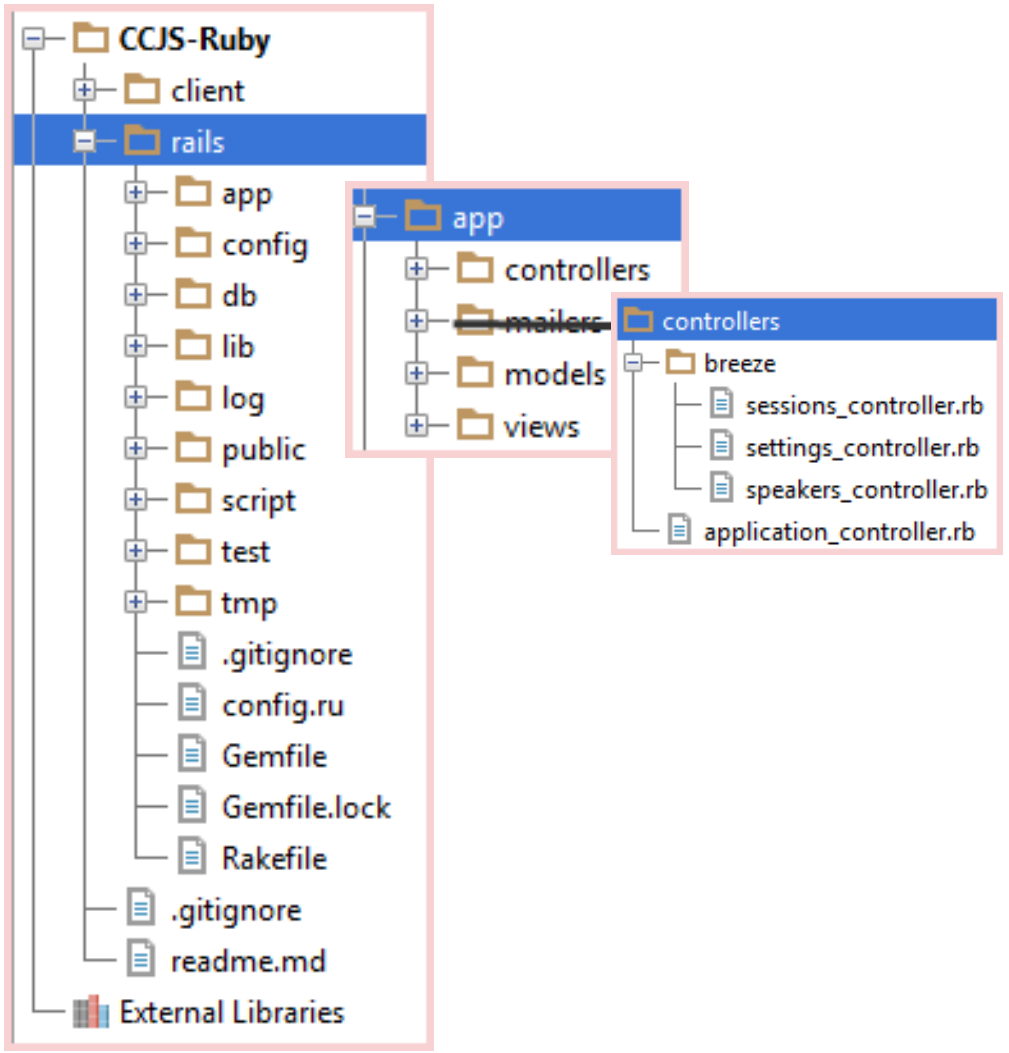
show.rabl

```
1  collection([@session], object_root: false)
2  node('$type') { 'CodeCamper.Sessions, CCJS.Model' }
3  attributes(*@session.attributes.keys)
```

update.rabl

```
1  object @session
2  attributes(*@session.attributes.keys)
```

RoR Server – Controllers



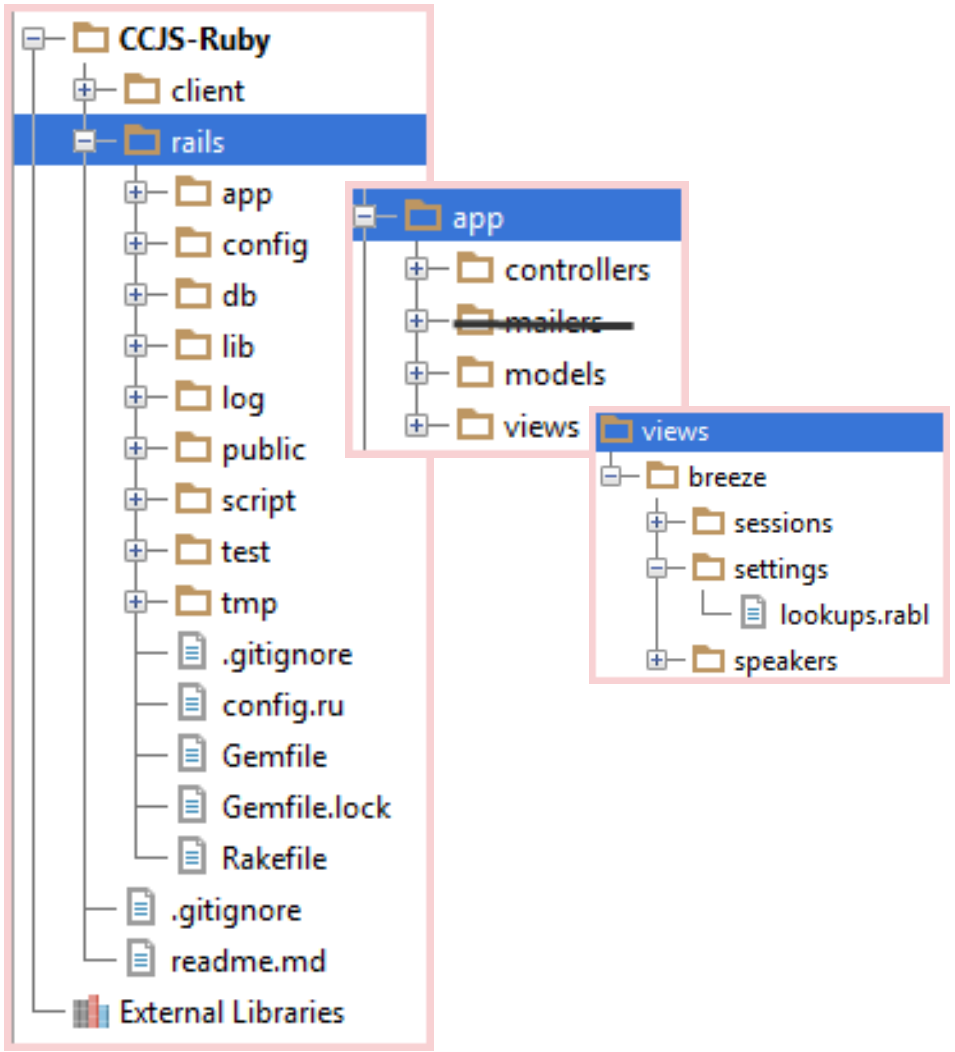
settings_controller.rb

```ruby
# Settings Controller Class
#
# Manage settings
class Breeze::SettingsController < ApplicationController

  # Metadata information
  def metadata
    render json: METADATA
  end

  # Rooms, Tracks and Time Slots information
  def lookups
    @rooms      = Room.all
    @tracks     = Track.all
    @time_slots = TimeSlot.all
  end

end
```

routes.rb

```ruby
RubyCCJS::Application.routes.draw do
  get '/breeze/Metadata', to: 'breeze/settings#metadata'
  get '/breeze/Lookups',  to: 'breeze/settings#lookups'
```

RoR Server - Views



lookups.rabl

```
1  node('$type') { 'CodeCamper.Lookups, CCJS.Model' }
2
3  child(@rooms, object_root: false) do
4    node('$type') { 'CodeCamper.Room, CCJS.Model' }
5    attributes :id, :name
6  end
7
8  child(@tracks, object_root: false) do
9    node('$type') { 'CodeCamper.Track, CCJS.Model' }
10   attributes :id, :name
11 end
12
13 child(@time_slots, object_root: false) do
14   node('$type') { 'CodeCamper.TimeSlot, CCJS.Model' }
15   attributes :id, :start, :duration, :is_session_slot
16 end
```

settings_controller.rb

```
11    # Rooms, Tracks and Time Slots information
12    def lookups
13      @rooms      = Room.all
14      @tracks     = Track.all
15      @time_slots = TimeSlot.all
16    end
17
18  end
```

API Differences

Rails serialization vs JSON.NET serialization

Rails Session

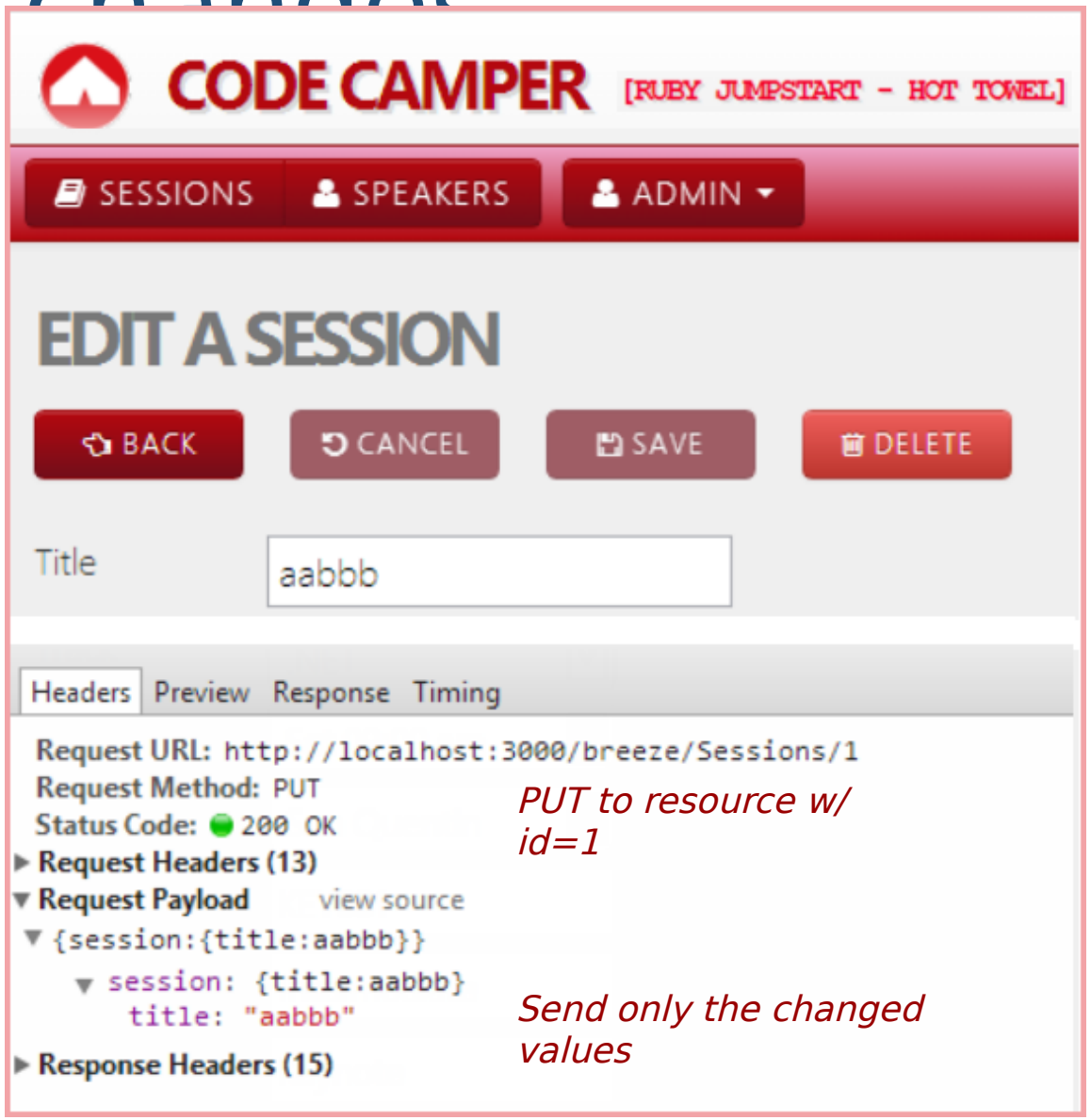
```
id: 1,
title: "Keynote",
code: "KEY001",
speaker_id: 10,
track_id: 5,
time_slot_id: 2,
room_id: 18,
level: "Intermediate",
tags: "Keynote",
$type: "CodeCamper.SessionPartials, CCJS.Model"
```

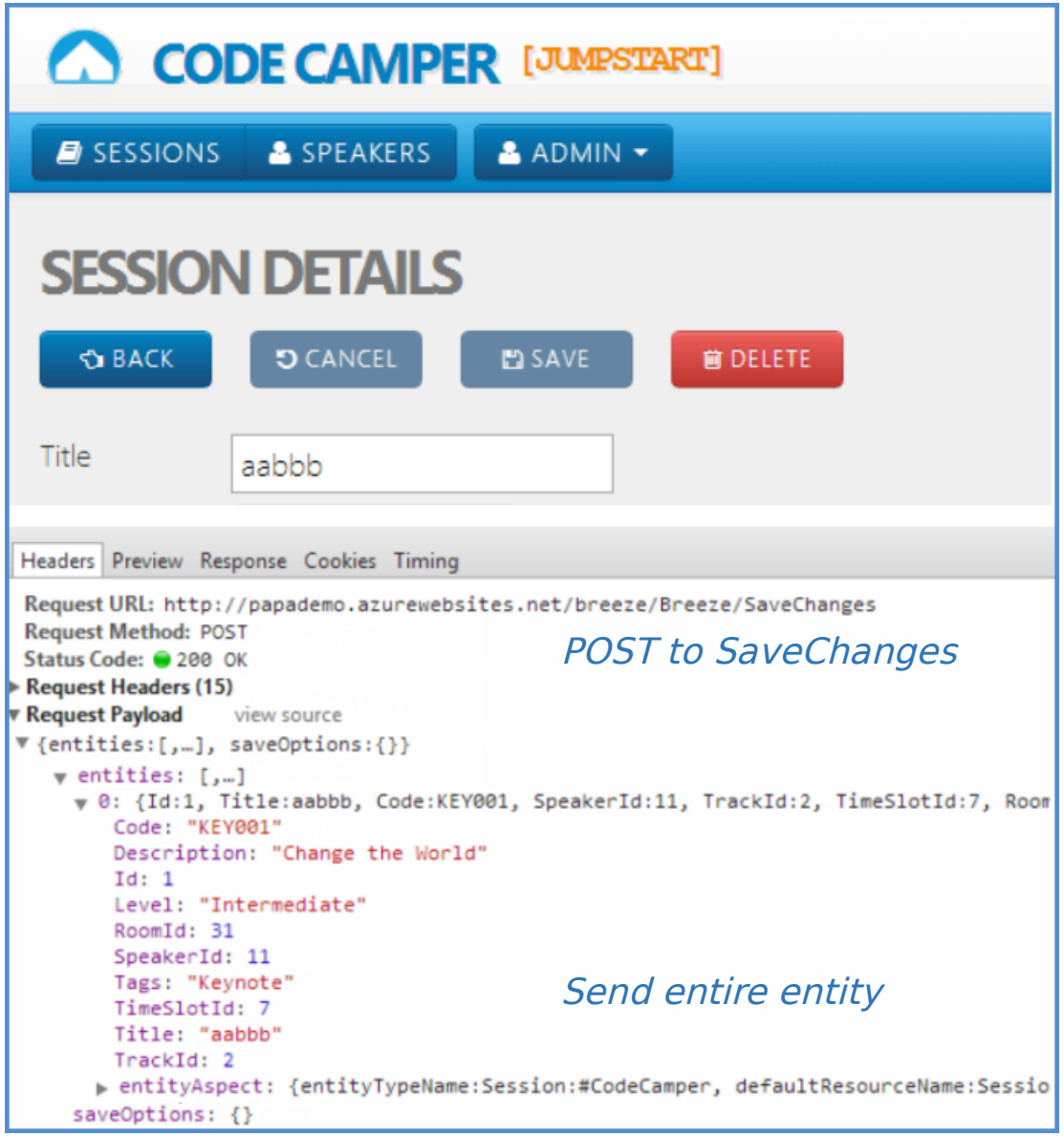
- *Rails-style entity property_names*
- *No $id node property*
- *$type node property is readable (v. anonymous type)*

JSON.NET Session

```
$id: "1",
$type: "_IB_DGU56or_prSk3yzZB87I8gCBWABk[[System.Int32, mscorlib],[System.String, mscorlib],[System.String, mscorlib],[System.Int32, mscorlib],
[System.Int32, mscorlib],[System.Int32, mscorlib],[System.Int32, mscorlib],[System.String, mscorlib],[System.String, mscorlib]],
_IB_DGU56or_prSk3yzZB87I8gCBWABk_IdeaBlade",
Id: 29,
Title: "Web Services at their Finest",
Code: "NET451",
SpeakerId: 13,
TrackId: 8,
TimeSlotId: 3,
RoomId: 32,
Level: "Advanced",
Tags: "WCF|REST|Web"
```

Rails REST update vs Breeze "save changes"

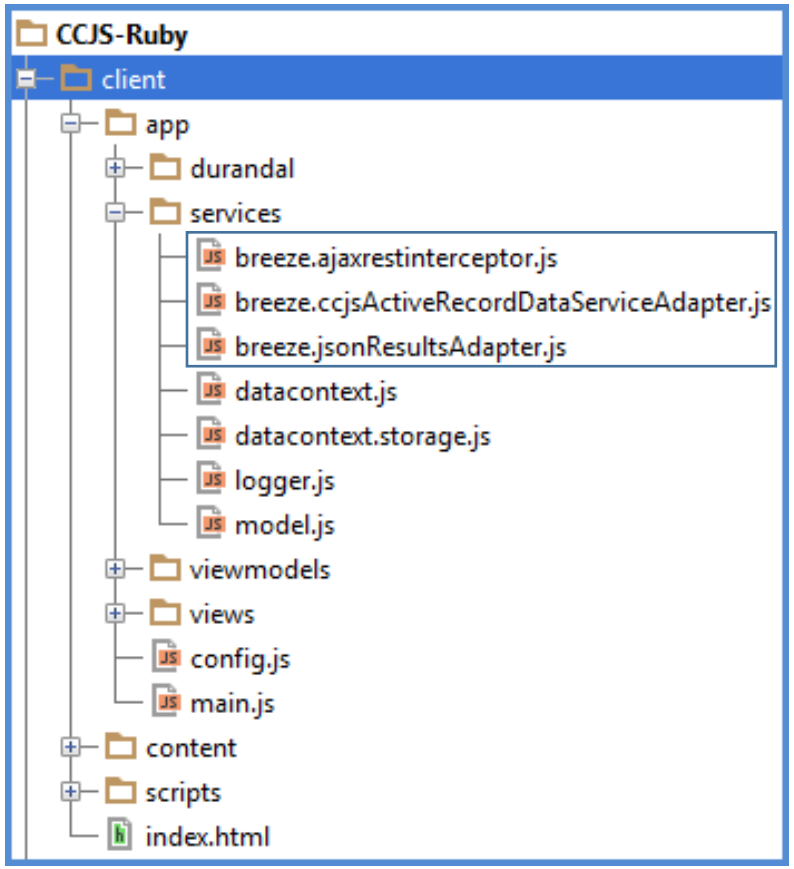




Adjust breeze client for Rails API

Configure adapters

Breeze extension points



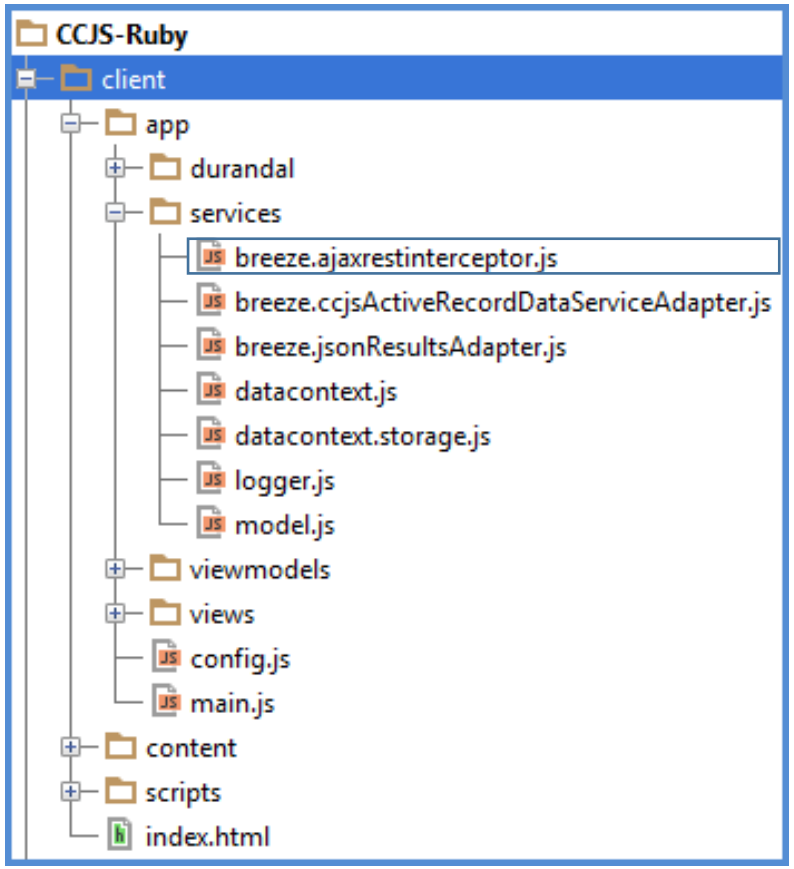
datacontext.js

```
1  define([
2      'durandal/system',
3      'services/model',
4      'services/datacontext.storage',
5      'config',
6      'services/logger',
7      'services/breeze.jsonResultsAdapter',
8      'services/breeze.ajaxrestinterceptor',
9      'services/breeze.ccjsActiveRecordDataServiceAdapter'],
10     function (system, model, DataContextStorage, config, logger, jsonResultAdapter) {
11
12         var manager = configureBreezeManager();
13
14         jsonResultAdapter.initialize(manager);
15
16         var ajaxInterceptor = new breeze.AjaxRestInterceptor();
17         ajaxInterceptor.enable();
18
19         breeze.config.initializeAdapterInstance("dataService", "ccjs_active_record", true);
```

Inject with RequireJS

AjaxRestInterceptor

Tweak the breeze ajax adapter to convert OData id-query into a REST URL



e.g., */breeze/Sessions/?$filter=id eq 1*

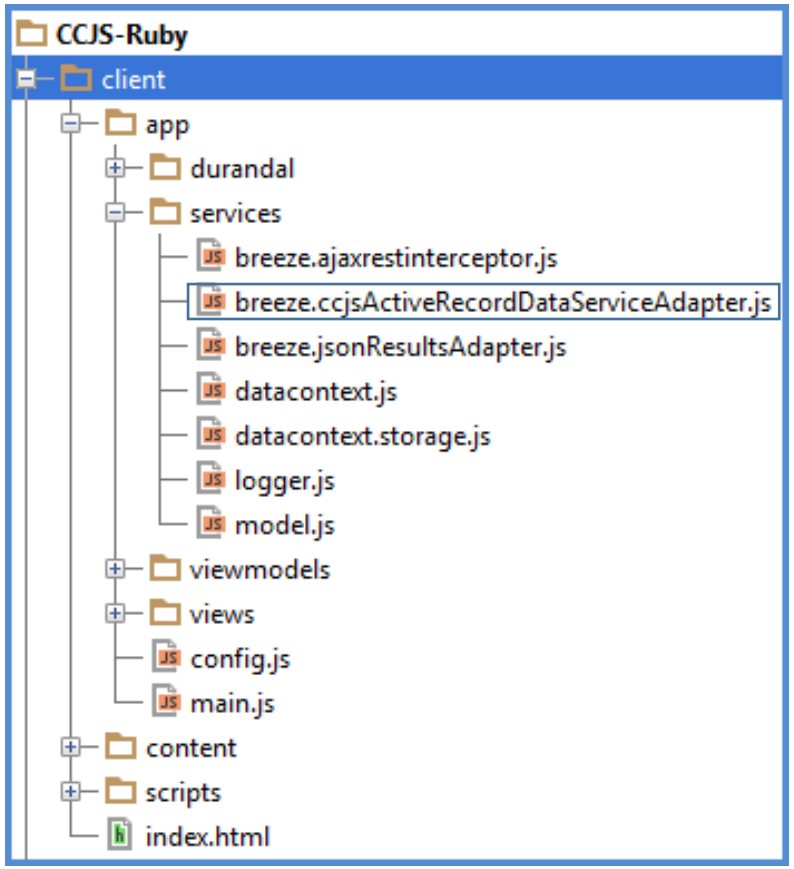
/breeze/Sessions/1

breeze.ajaxrestinterceptor.js

```
16  breeze.AjaxRestInterceptor = (function () {
17      /**
18       Wraps the ambient breeze ajax adapter's `ajax` method with an interceptor
19       that converts certain URLs into a more "ReSTy" design.
20
21       Ex: Converts '/breeze/orders/?$filter=id eq 1' into '/breeze/orders/1'.
22
23       After instantiating the adapter, call its enable() method to enable its injection into th
24       base ajax adapter. Call its disable() method to restore the pre-injection behavior.
25
26       **/
```

DataServiceAdapter

Replace POST to “SaveChanges” with REST

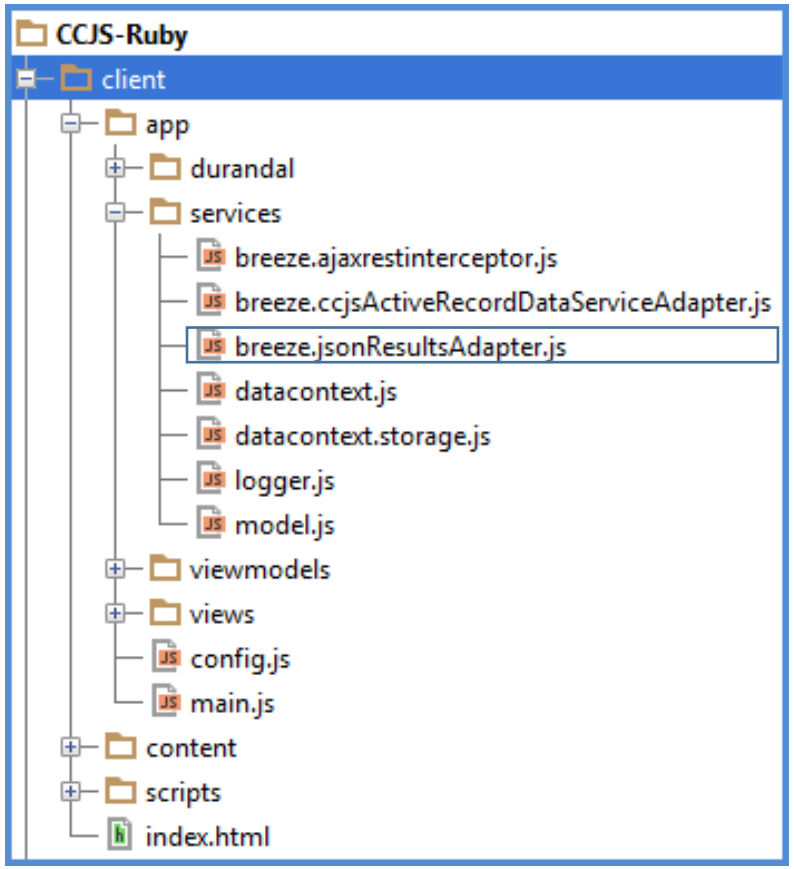


breeze.ccjsActiveRecordDataServiceAdapter.js

```
60  ctor.prototype.saveChanges = function(saveContext, saveBundle) {
61
62      var requestInfo;
63      var deferred = Q.defer();
64      var that = this;
65
66      try{
67          requestInfo = this._getRequestInfo(saveContext, saveBundle);
68      } catch (err) {
69          deferred.reject(err);
70          return deferred.promise;
71      }
72
73      // Right now can only make a single save request
74      // for a single entity
75      var request = {
76          method: requestInfo.method,
77          url: requestInfo.url,
78          data: requestInfo.data,
79          dataType: 'json',
80          accept:  {json:'application/json'},
81          contentType: 'application/json; charset=utf-8',
82          success: saveSuccess,
83          error: saveFail
84      }
85
86      ajaxImpl.ajax(request);
87
88      return deferred.promise;
```

JsonResultsAdapter

Identify “partial entity” JSON data nodes

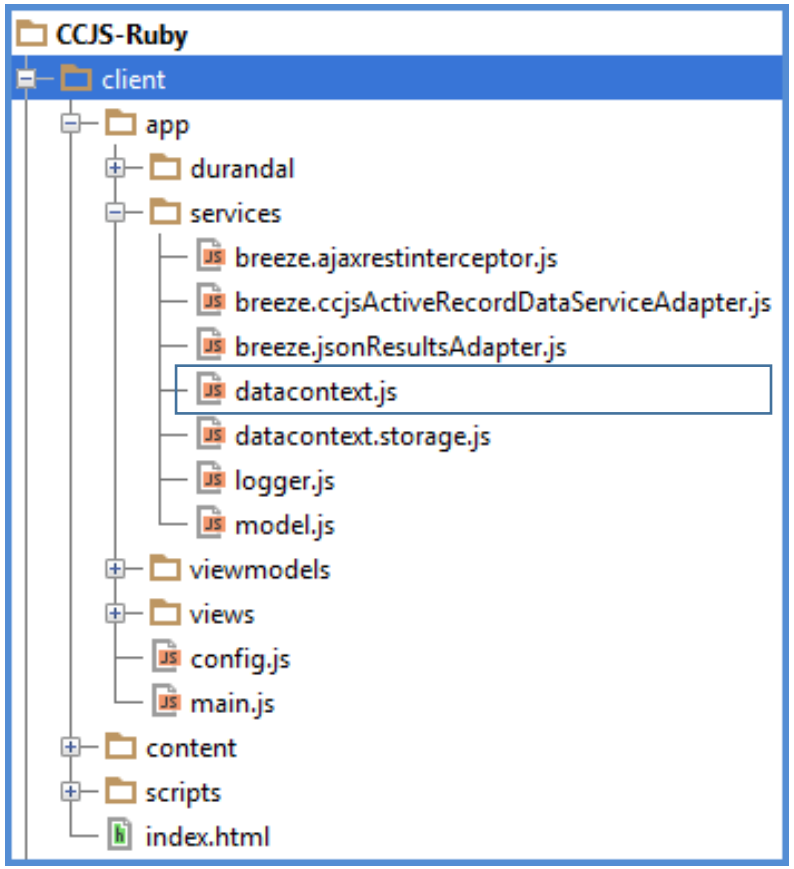


breeze.jsonResultsAdapter.js

```
16  // Extend the visitNode method of a source JsonResultsAdapter
17  // to set Session.isPartial and Person.isPartial to true
18  // when a query returns partial entity data.
19  function createCustomAdapter(sourceAdapter) {
20
21      var baseVisitNode = sourceAdapter.visitNode;
22
23      var visitNode = function (node, mappingContext, nodeContext) {
24          // With .NET server the  projected typename is unpronouncable
25          // In Rails version where we control the projected type name.
26          // if type name contains 'partial' .. , it's a partial entity
27          node.isPartial =  /partial/i.test(node.$type)
28          return baseVisitNode(node, mappingContext, nodeContext);
29      };
30
31      return new breeze.JsonResultsAdapter({
32          name: 'ccjs',
33          visitNode: visitNode
34      });
35  }
```

NamingConvention

Convert property names
e.g., *timeSlotId* □□
time_slot_id



datacontext.js

```
250  function configureBreezeManager() {
251
252      var rubyNamingConvention = new breeze.NamingConvention({
253          name: "rubyNamingConvention",
254          serverPropertyNameToClient: function (serverPropertyName) {
255              var serverPropertyNameArray = serverPropertyName.split('_')
256              clientPropertyName = serverPropertyNameArray.shift();
257              $.each(serverPropertyNameArray, function(i, v) { clientPropertyName += v.charAt(0
258              return clientPropertyName;
259          },
260          clientPropertyNameToServer: function (clientPropertyName) {
261              var serverPropertyName = clientPropertyName.replace(/([a-z])([A-Z])/g, '$1_$2').t
262              return serverPropertyName;
263          }
264      });
265
266      rubyNamingConvention.setAsDefault();
```

Thank you, RubyTribe