Building a Data Access Layer for Your Data Using Sinatra + Ruby + MySQL

Matt Yonkovit

Percona Live London 2011

The dream of developing fast, efficient, applications without detailed knowledge of the database....

Way Back Machine

- Old school database design
  - Business logic goes into stored procedures
  - Developers do not touch the tables directly
  - Many applications can access the same procs, reusing code
  - Applications can be somewhat modular

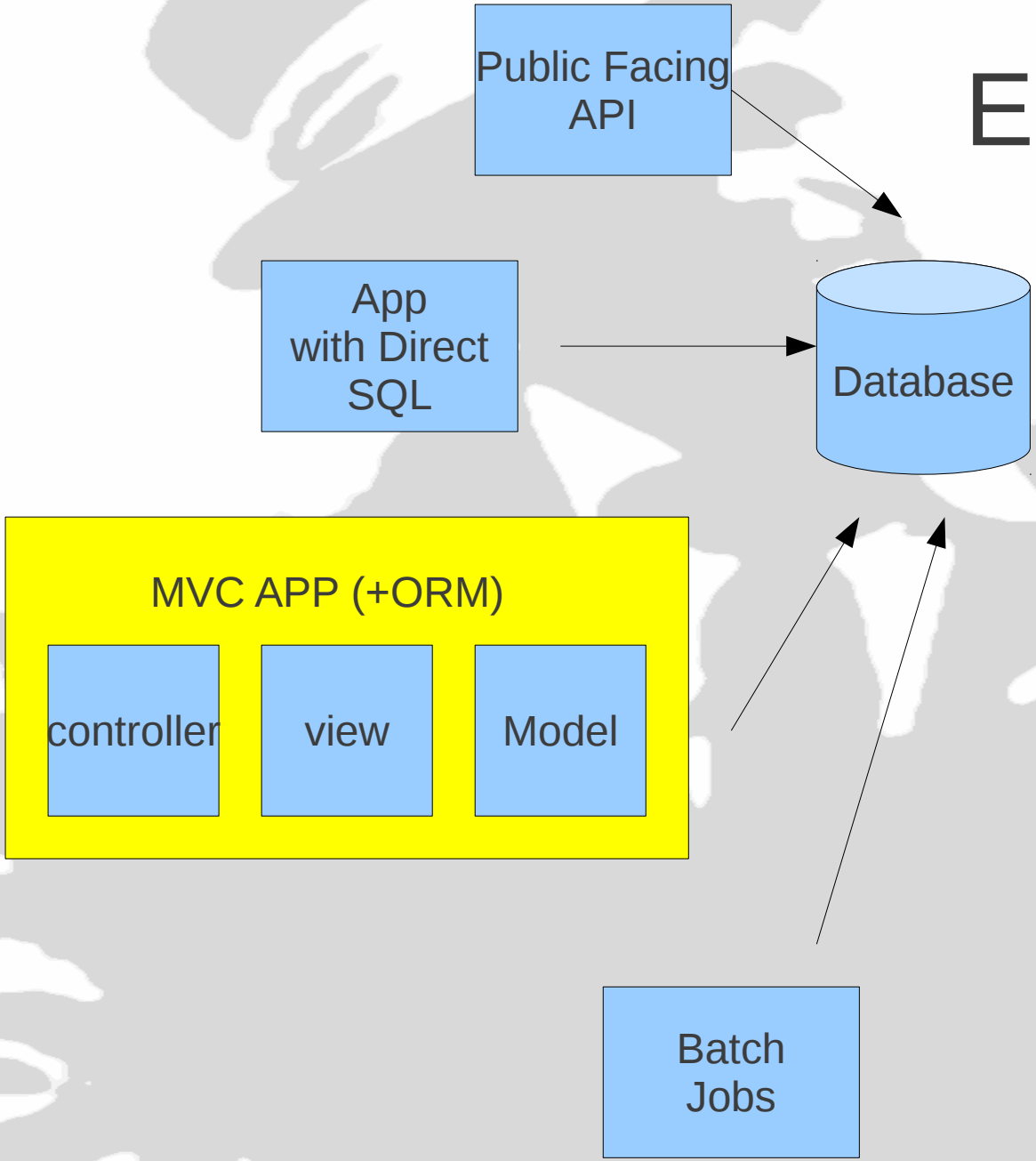
The rise of ORM's

- Object Relational Mapping tools are still very popular
  - They hide the complexity of the database
  - They make data interactions seem intuitive in code
  - They generate sql for you
- They have issues
  - Generated SQL is often less then optimal
  - Many times models are not shared across applications

Development Today

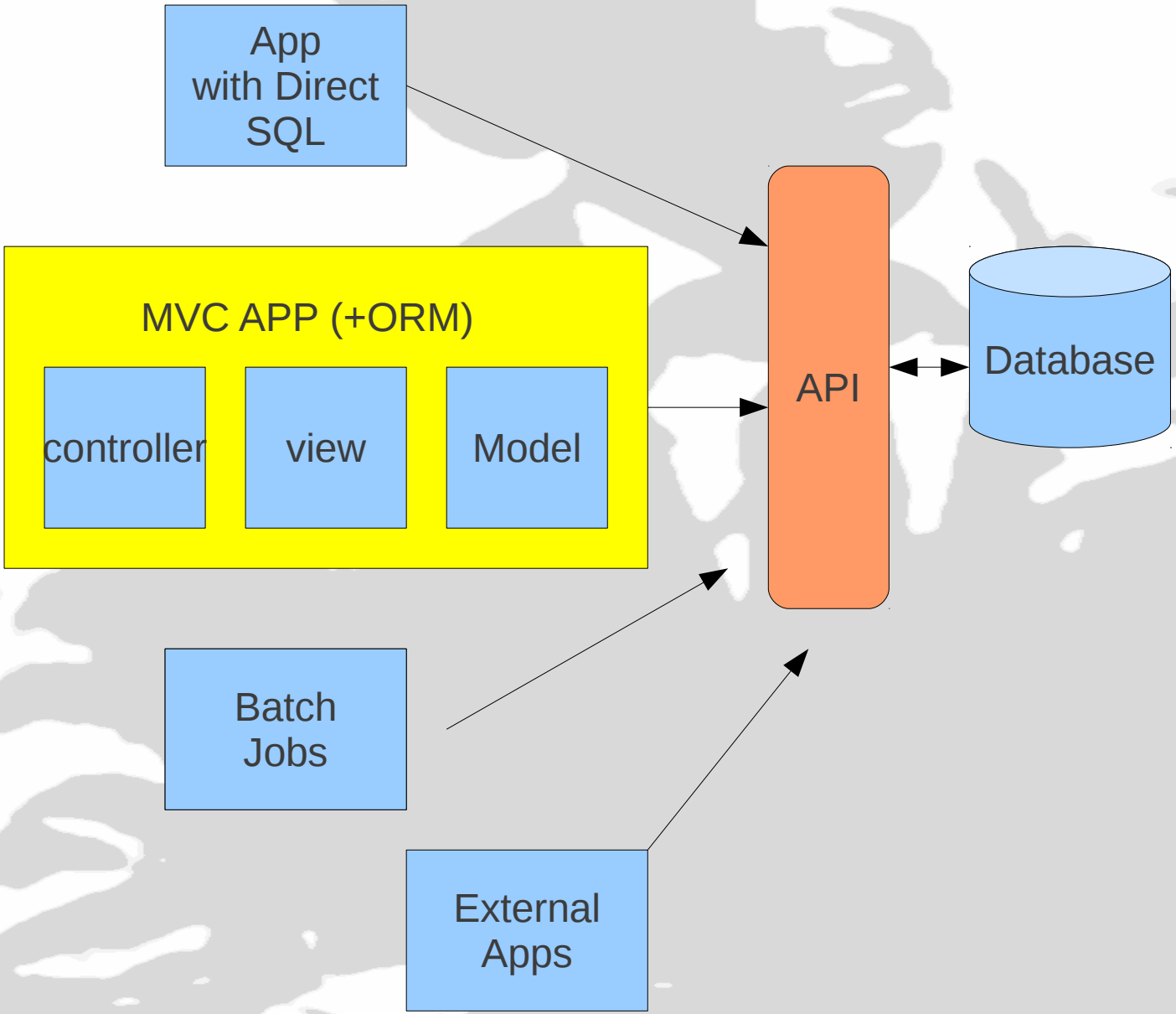
- Many people have the following already
  - Web Services
  - A web front-end/ui
  - Make heavy use of Javascript
  - Have lots of data to display
  - Have more then 1 application using the same database

Complex Environments



- Database changes cascade to multiple applications
- Business logic built for 1 application often is duplicated or ignored in another application

Modular design



What is Sinatra?

A really tiny web framework for building apps in Ruby.

Read all about it:
http://www.sinatrarb.com/
http://sinatra-book.gittr.com/

How To Get started

- Linux Machine
  - RVM (Ruby Version Manager) http://beginrescueend.com/
  - Ruby installed (any version is ok, I prefer 1.9.2)
  - Your favorite database driver/connection library
    - Mysql gem, mysql2 gem, active record, sequel, etc.
  - Other gems depending on your needs:
    - Xmlsimple
    - Json
    - etc

Hello World

Api.rb

```
require 'rubygems'
require 'sinatra'

get '/' do
  "Hello world, it's #{Time.now} at the server!"
end
```

How do you run it?

```
>>ruby Api.rb -p 9000
```

Point your web browser to localhost:9000

Thats it!

Demo of Hello World

Returning MySQL Data

```
require 'rubygems'
require 'mysql2'
require 'sinatra'


@@mysqlclient = Mysql2::Client.new(:host => "localhost", :username => "root", :database => "information_schema")


get '/' do
  "Hello world, it's #{Time.now} at the server!"
end

get '/tables/' do
   res = Array.new
   result = @@mysqlclient.query("SELECT * FROM tables",:as => :array)
   result.each do | row |
         res.push(row)
   end
   return res.to_s
end
```

Demo: Getting MySQL Data Back

Can do the Same Thing With AR

```
require 'rubygems'
require 'mysql2'
require 'active_record'
require 'sinatra'

get '/' do
  "Hello world, it's #{Time.now} at the server!"
end

get '/tables/' do
   MyAPI.get_tables()
end


class MyAPI < ActiveRecord::Base
  class << self

   dbconfig = {
    :adapter  => "mysql2",
    :host     => "localhost",
    :username => "root",
    :password => "",
    :database => "information_schema"
    }
   ActiveRecord::Base.establish_connection(dbconfig)
```

```
  def get_tables()
     query='select * from tables';
     self.connection.select_all(query).to_s
  end

end
end
```

Demo: Getting MySQL Data Back

Format the Data!

```
require 'rubygems'
require 'mysql2'
require 'active_record'
require 'sinatra'
require 'csv'
require 'sinatra/respond_to'
require 'json'
require 'xmlsimple'

Sinatra::Application.register Sinatra::RespondTo

get '/' do
  "Hello world, it's #{Time.now} at the server!"
end

get '/tables' do
   format_response MyAPI.get_tables()
end

def format_response (package)
    respond_to do |wants|
    wants.txt { package.to_s }
    wants.csv { package.to_csv }
    wants.xml { package.to_xml }
    wants.json {  package.to_json  }
    end
end
```

```
class MyAPI < ActiveRecord::Base
  class << self

   dbconfig = {
    :adapter  => "mysql2",
    :host     => "localhost",
    :username => "root",
    :password => "",
    :database => "information_schema"
    }
   ActiveRecord::Base.establish_connection(dbconfig)


  def get_tables()
      query='select * from tables';
      self.connection.select_all(query)
  end

end
end
```

Demo: Json/XML Formatting!

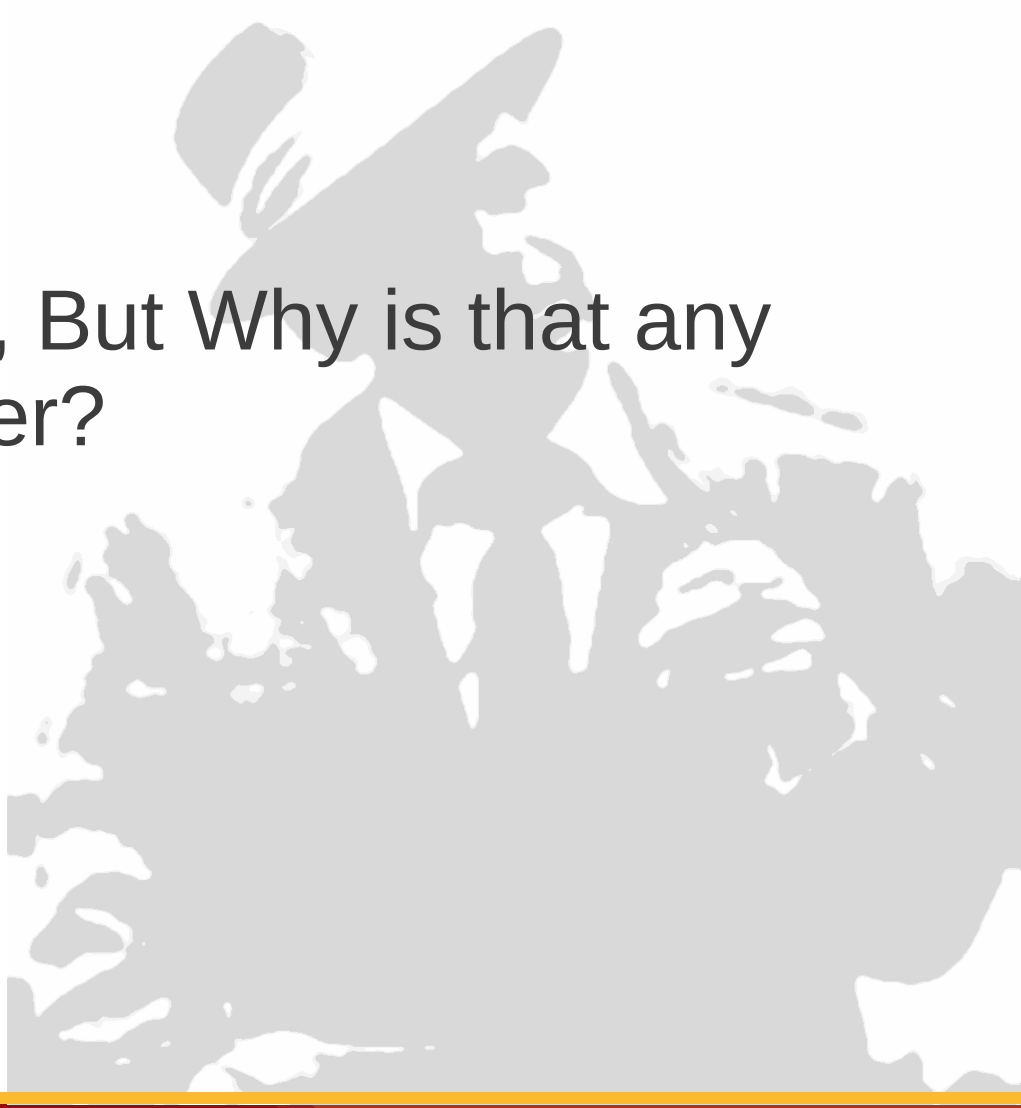
Filtering Data

```
get '/tables/?:schema?/?:table?' do
   format_response MyAPI.get_tables(params)
end
```

```
def get_tables(params)
      where = "where 1=1"
      if ! params[:schema].nil?
          where = "#{where} and TABLE_SCHEMA = '#{params[:schema]}'"
      end
      if ! params[:table].nil?
          where = "#{where} and TABLE_NAME = '#{params[:table]}'"
      end
      query="select * from tables #{where}";
      if params[:no_header]
         self.connection.select_rows(query)
      else
        self.connection.select_all(query)
      end
end
```

Demo: Data Filtering!

Thats All Great and all, But Why is that any Easier?

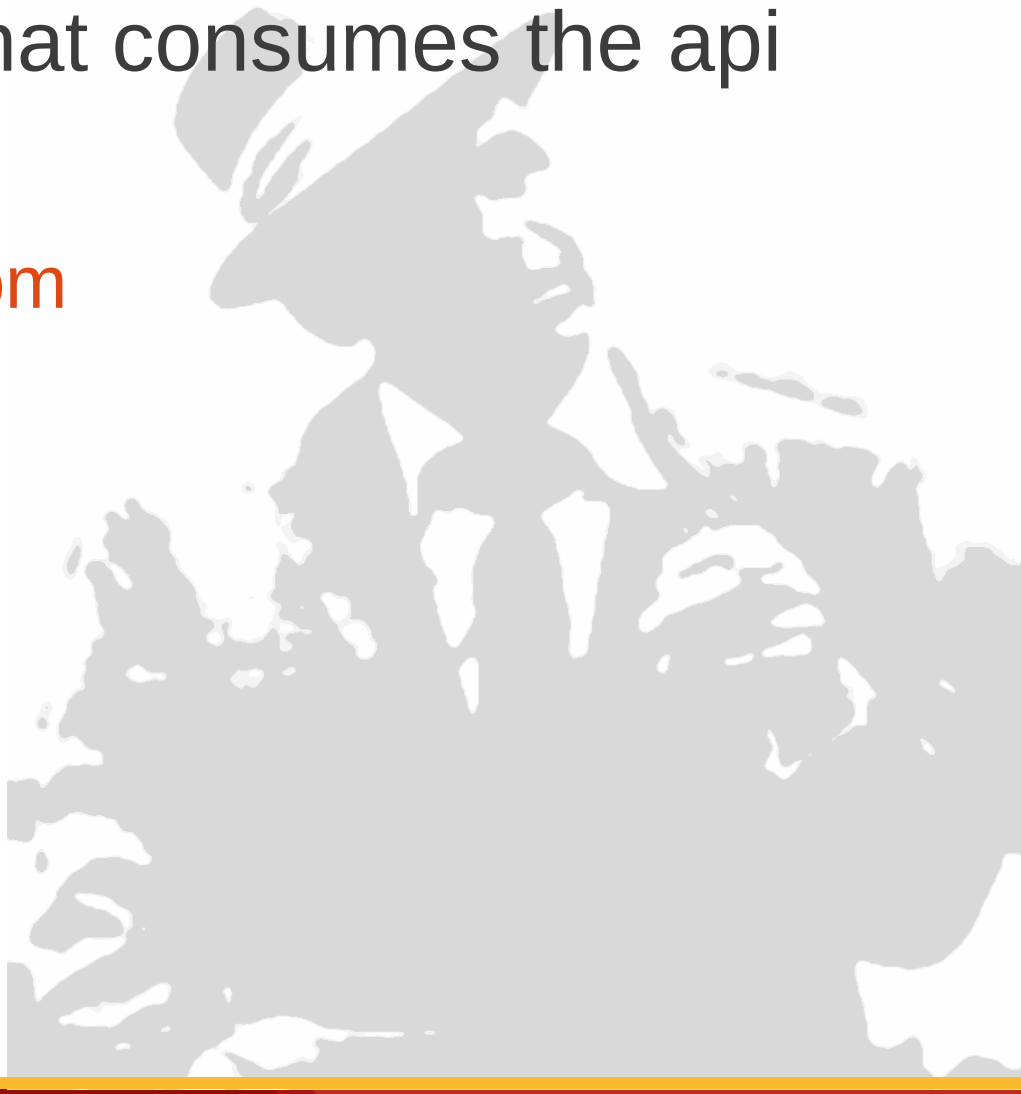


Consuming the API

- JSON can be consumed and used natively in many different programing languages
- You can build very feature rich Javascript only applications
- Many popular javascript graphing libraries can easily consume and graph data passed back from the API

Demo: Putting It Together

- Javascript application that consumes the api
  - http://datatables.net/
  - http://www.highcharts.com



Thank You to Our Sponsors

Platinum Sponsor

Clustrix

Gold Sponsor

FUSION-IO

Silver Sponsors

adfonic

continuent
Open. Always Available.

facebook

GENIEDB

several nines

SkySQL

SCHOONER
INFORMATION TECHNOLOGY

Percona Live London Sponsors

Exhibitor Sponsors

Friends of Percona Sponsors

Media Sponsors

Annual MySQL Users Conference

Presented by Percona Live

The Hyatt Regency Hotel, Santa Clara, CA

April 10th-12th, 2012

Featured Speakers

Mark Callaghan, Facebook

Jeremy Zawodny, Craigslist

Marten Mickos, Eucalyptus Systems

Sarah Novotny, Blue Gecko

Peter Zaitsev, Percona

Baron Schwartz, Percona

The Call for Papers is Now Open!

Visit www.percona.com/live/mysql-conference-2012/

baron@percona.com
@xaprb

We're Hiring! www.percona.com/about-us/careers/

PERCONA LIVE

www.percona.com/live