Fork me on GitHub

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Databases ¶ ↑

Often times, applications need to talk to some external datasource. This section will cover recipes for integrating different types of databases into your Sinatra application. This section won't cover the differences between the types of databases your application may want to use, it will just list a few recipes for connecting to and using data in some of those databases. For a discussion of the different types of databases out there you can consult this Wikipedia article.

- Mongo

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Fork me on GitHub

Sinatra Recipes

Community contributed recipes and techniques

Select a topic

Chapters

Mongo ¶ ↑

Mongo is a document-oriented database. Though Object Relational Mappers (ORMs) are often used to connect to databases, you will see here that it is very easy to connect your applications to a Mongo database without the use of an ORM, though several exist. See the Mongo models page for a discussion of some of the ORMs available.

Install the required gems:

```
gem install mongo
gem install bson_ext
```

The first step is in connecting your application to an instance of Mongo is to create a connection. You can do this in your *configure* block:

Note that there has been a change in the Ruby API post v 1.8.x. The following examples use the newer API

```
require 'rubygems'
require 'sinatra'
require 'mongo'
require 'json/ext' # required for .to_json

include Mongo

configure do
  conn = MongoClient.new("localhost", 27017)
  set :mongo_connection, conn
  set :mongo_db, conn.db('test')
end
```

With a connection “in hand” you can connect to any database and collection in your Mongo instance.

```
get '/collections/?' do
  settings.mongo_db.collection_names
end

helpers do
  # a helper method to turn a string ID
  # representation into a BSON::ObjectId
  def object_id val
    BSON::ObjectId.from_string(val)
  end

  def document_by_id id
    id = object_id(id) if String === id
    settings.mongo_db['test'].
      find_one(:_id => id).to_json
  end
end
```

Finding Records¶ ↑

```
# list all documents in the test collection
get '/documents/?' do
  content_type :json
  settings.mongo_db['test'].find.to_a.to_json
end

# find a document by its ID
get '/document/:id/?' do
  content_type :json
  document_by_id(params[:id]).to_json
end
```

Inserting Records¶ ↑

```
# insert a new document from the request parameters,
# then return the full document
post '/new_document/?' do
  content_type :json
  new_id = settings.mongo_db['test'].insert params
  document_by_id(new_id).to_json
end
```

Updating Records¶ ↑

```
# update the document specified by :id, setting its
# contents to params, then return the full document
put '/update/:id/?' do
  content_type :json
  id = object_id(params[:id])
  settings.mongo_db['test'].update(:_id => id, params)
  document_by_id(id).to_json
end

# update the document specified by :id, setting just its
# name attribute to params[:name], then return the full
# document
put '/update_name/:id/?' do
  content_type :json
  id   = object_id(params[:id])
  name = params[:name]
  settings.mongo_db['test'].
    update(:_id => id, {"$set" => {:name => name}})
  document_by_id(id).to_json
end
```

Deleting Records¶ ↑

```
# delete the specified document and return success
delete '/remove/:id' do
  content_type :json
  settings.mongo_db['test'].
    remove(:_id => object_id(params[:id]))
  {:success => true}.to_json
end
```

For more information on using the Ruby driver without an ORM take a look at MongoDB's tutorial.

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top

Sinatra Recipes

Community contributed recipes and techniques

Fork me on GitHub

Select a topic

Chapters

MongoDB ¶ ↑

There are many ORMs out there for Mongo (or *ODMs* in this case). This page will go over just a few.

Mongo ¶ ↑

Looking for the official driver?

MongoMapper ¶ ↑

```
require 'mongomapper'
```

Create the Model class

```
class Link
  include MongoMapper::Document
  key :title, String
  key :link, String
end
```

Create the route:

```
get '/' do
  @links = Link.all
  haml :links
end
```

Mongoid ¶ ↑

```
require 'mongoid'
```

Create the Model class

```
class Link
  include Mongoid::Document
  field :title, :type => String
  field :link, :type => String
end
```

Create the route:

```
get '/' do
  @links = Link.all
  haml :links
end
```

Candy ¶ ↑

```
require 'candy'
```

Create the Model class

```
class Link
  include Candy::Piece
end

class Links
  include Candy::Collection
  collects :link   # Declares the Mongo collection is 'Link'
end

Link.connection # => Defaults to localhost port 27017
Link.db         # => Defaults to your username, or 'candy' if unknown
Link.collection # => Defaults to the class name ('Link')
```

Create the route:

```
get '/' do
  @links = Links.all
  haml :links
end
```

Mongomatic ¶ ↑

```
require 'mongomatic'
```

Create the Model class

```
class Link < Mongomatic::Base
  def validate
    self.errors.add "title", "blank" if self["title"].blank?
    self.errors.add "link",  "blank" if self["link"].blank?
  end
end
```

Create the route:

```
get '/' do
  @links = Link.all
  haml :links
end

def validate
  self.errors.add "name", "blank" if self["name"].blank?
  self.errors.add "email", "blank" if self["email"].blank?
  self.errors.add "address.zip", "blank" if (self["address"] || {})["zip"].blank?
end
```

MongoODM ¶ ↑

```
require 'mongo_odm'
```

Create the Model class

```
class Link
  include MongoODM::Document
  field :title
  field :link
end
```

Create the route:

```
get '/' do
  @links = Link.find.to_a
  haml :links
end
```

GitHub links¶ ↑

- MongoMapper
- Mongoid
- Candy
- Mongomatic
- MongoODM

Did we miss something?

It's very possible we've left something out, that's why we need your help!This is a community driven project after all. Feel free to fork the project and send us a pull request to get your recipe or tutorial included in the book.

See the README for more details.

Top