In this lecture, we will discuss…

- Introduction to Mongoid
- Mongoid installation
- Mongoid Configuration

Mongoid

- Mongoid (pronounced mann-goyd) - Object-Document-Mapper (ODM) for MongoDB written in Ruby
- Mix of Active Record and MongoDB's schema-less and performant document-based design
- Dynamic queries, and atomic modifier operations

Mongoid - Installation

- Add Mongoid to your `Gemfile`
  - `gem 'mongoid', '~> 5.0.1'`

Mongoid Configuration

- `rails g mongoid:config`
- generates a config file
- Edit - appname/config/mongoid.yml
- The YML file generates
  - appname_development
  - appname_test

Summary

- Mongoid – Object Document Mapper (ODM)
- Providing familiar API for Ruby developers (ActiveRecord)

What's Next?

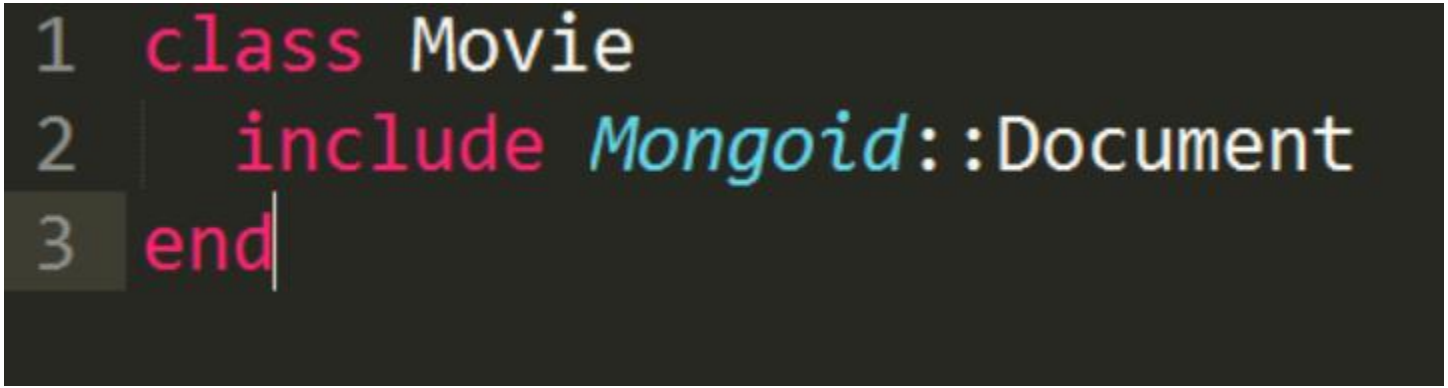
- Document and Custom class

In this lecture, we will discuss…

- ✧ Document Class
- ✧ Fields
- ✧ Field Types
- ✧ Field Aliases
- ✧ Custom Fields

Document

- Documents are the core objects in Mongoid
  - `Mongoid::Document`
- Documents can be stored in a collection or embedded in other documents



Fields

- ✧ Fields are attributes
  - `field`
  - `type` - String by default
  - `rails g model`

```
1 class Movie
2   include Mongoid::Document
3   field :title, type: String
4   field :type, type: String
5   field :rated, type: String
6   field :year, type: Integer
7 end
```

`rails g model` - command

```
$ rails g model Movie title type rated year:integer release_date:date \
    runtime:Measurement votes:integer countries:array languages:array \
    genres:array filming_locations:array metascore simple_plot:text \
    plot:text url_imdb url_poster directors:array actors:array
```

```
$ rails g model Actor name birth_name data_of_birth:Date height:Measurement bio:text
```

Field Types

Array	Boolean	DateTime	Hash
BigDecimal	Date	Float	Integer
BSON	Range	Regexp	String
Symbol	Time	TimeWithZone	

Timestamps

- Timestamp information is not added by default in Mongoid -- as it is within ActiveRecord.
- `created_at` and `updated_at` fields done via `Mongoid::Timestamps` mixin.
- `touch` - Will update the document's `updated_at` timestamp.

Field Aliases

```ruby
class Actor
  include Mongoid::Document
  include Mongoid::Timestamps

  field :name, type: String
  field :birthName, as: :birth_name, type: String
  field :data_of_birth, type: Date
  field :height, type: Measurement
  field :bio, type: String
```

- birthName in document → mapped to birth_name in model
- Comply with rails naming convention
- Helps during compression

Custom Fields

- You can define custom types in Mongoid and determine how they are serialized and deserialized
- 5 methods in total
  - initialize
  - mongoize (instance method)
  - mongoize, demongoize, evolve (class methods)
- Example: `Measurement`

```
:runtime=>{:amount=>60, :units=>"min"}
```

store_in

```
1 class Location
2     include Mongoid::Document
3     store_in collection: "places"
4     field :city, type: String
5     field :state, type: String
6     field :country, type: String
7 end
```

- Application type to Document type mapping
- `Location` gets stored in to “`places`” collection

Summary

- ✧ Good data type support
- ✧ Aliases, Timestamps
- ✧ Document class and Custom Fields

What's Next?

- ✧ CRUD

In this lecture, we will discuss…

- ✧ `create`
- ✧ `find`
- ✧ `update`
- ✧ `upsert`
- ✧ `delete`

Operation – Model.create

- Model.create

```
movie = Movie.create(
  title: "Martian",
  type: "Thriller",
  rated: "R",
  year: 2015
)
```

- Will insert a document into “movies” collection
  - Movie will automatically create a collection called movies (if ‘movies’ does not exist)

Operation – Model (save)

```
movie = Movie.new(
  title: "Rocky",
  type: "Action",
  rated: "R",
  year: 1975
)
movie.save
```

Saves the changed attributes to the database automatically

- Update
  - `movie.year = 1986`
  - `movie.save`

Operation – update_attributes

```
movie = Movie.new(
   title: "Rocky31",
   rated: "PG-13"
)

movie.update_attributes(:rated => "R")
```

Operation – Model#upsert

- Performs a MongoDB `upsert` on the document
- If document exists – will get overwritten
- If document does not exists – will get inserted
  - ```
    movie = Movie.create(
      title: "Rocky31",
      rated: "PG-13"
    )
    ```
  - ```
    Movie.new(:_id=>"<ID>",
    :title=>"Rocky31", :rated=>"R").upsert
    ```

Operation – Model#delete

- `movie.delete`
  - will delete the document in the database
- `Movie.delete_all`
  - will delete all documents from the ‘movies’ collection

Summary

- ✧ Mongoid supports all expected CRUD operations

What's Next?

- ✧ Movie application setup

In this lecture, we will discuss…

- ✧ Setup
- ✧ Initialization
- ✧ Model class
- ✧ Custom class

Setup – Data files

- movies.json
- actors.json
- directors.json
- writers.json
- places.json

Initialization

- Import data
  - `> rake db:seed`
- Setup index
  - `> rake db:mongoid:create_indexes`

Model Types and Document Representation

- ✧ Custom Types
  - `Measurement` – represents measurement info
  - `Point` – represents geolocation points

Model Types and Document Representation

✧ Document Model Class

- `Place` – abstraction added to `Point` to hold location information about the geolocation point
- `Actor` - represents someone who plays a role in a `Movie`
- `Writer` – one of the authors of a `Movie`
- `Director` – one of the directors of a `Movie`
- `Movie` – core information about a `Movie`

Model Types and Document Representation

✧ Document Model Class

- `DirectorRef` – is an annotated reference to a `Director`
  - used to cache stable/core director information that the referencing document/view will need.
- `MovieRole` - is a character in a `Movie` played by an `Actor`

Relationships - Types

- 1:1 Embedded (Actor -> place_of_birth:Place)
- M:1 Linked (Director -> residence:Place)
- 1:M Embedded (Movie <-> roles:MovieRole)
- M:1 Embedded Linked (MovieRole <-> Actor)
- 1:1 Linked (Movie -> sequel_to:Movie)
- M:M (Movie <-> writers:Writer)

Summary

- Models are key as relations are associations between one model and another in the domain and in the database.

What's Next?

- Relationship Types