In this lecture, we will discuss…

- Constraints
- Validation
- Dependent behavior
  - `:delete`
  - `:destroy`
  - `:nullify`
  - `:restrict`

Field Validation

- ActiveModel validations can be added to Mongoid model classes.

```
1 class Director
2   include Mongoid::Document
3   include Mongoid::Timestamps
4   field :name, type: String
5       .........
6   validates_presence_of :name
7 end
```

“***name is mandatory***”

Dependent Behavior

- Mongoid supports dependent options to manage referenced associations
- Will instruct Mongoid to handle delete situations
- `:delete, :destroy, :nullify, :restrict`

Relationship Constraints

- `(default)` – Orphans the child document
  - 1:1 and 1:M leaves the child with stale reference to the removed parent
  - M:M clears the child of the parent reference (acts like `:nullify`)
- `:nullify` – Orphans the child document after setting the child foreign key to nil

Relationship Constraints

- `:destroy` – Remove the child document after running model callbacks on the child
- `:delete` – Remove the child document without running model callbacks on the child
  - M:M does not remove the child document from database (acts like :`nullify`)
- `:restrict`– Raise an error if a child references the parent being removed

delete vs destroy

- `:delete` – will only delete current object – straight delete in the database.
- `:destroy`- will delete current object and associated children record - analyzes the class you're deleting, determines what it should do for dependencies, runs through validations (callbacks - `:before_destroy`, `:after_destroy`.)

Callbacks – Movie Model

```
before_destroy do |doc|
  puts "before_destroy Movie callback for #{doc.id}, "\
       "sequel_to=#{doc.sequel_to}, writers=#{doc.writer_ids}"
end
after_destroy do |doc|
  puts "after_destroy Movie callback for #{doc.id}, "\
       "sequel_to=#{doc.sequel_to}, writers=#{doc.writer_ids}"
end
```

Callbacks – Writer Model

```
before_destroy do |doc|
  puts "before_destroy Writer callback for #{doc.id}, "\
       "movies=#{doc.movie_ids}"
end
after_destroy do |doc|
  puts "after_destroy Writer callback for #{doc.id}, "\
       "movies=#{doc.movie_ids}"
end
```

DEMO

Script – Demo Setup

- `reload!`
- `rocky30=Movie.create(:title=>"Rocky 30")`
- `rocky31=Movie.create(:title=>"Rocky 31", :sequel_to=>rocky30)`
- `writer=rocky30.writers.create (:name=>"A Writer")`

Script – Data Cleanup

```
p Movie.where(:title=>
   {:$regex=>"Rocky 3[0-1]"}).delete_all; \

Writer.where(:name=>{:$regex=>"Writer"})
     .delete_all
```

Summary

- Constraints, Validations and Dependent Behavior – key in maintaining the state of your application data.

What's Next?

- Queries