In this lecture, we will discuss…

- ✧ Resources - Standalone and Dependent
- ✧ Using `rails` to build resources
- ✧ Example Resources
  - `Movie`
  - `Actor`
  - `MovieRole`

Resource Scope

- Resource - fundamental concept in any RESTful API
  - *is an object with a type, associated data, relationships to other resources, and a set of methods that operate on it.*

- Example Resources
  - Movies
  - Actors
  - MovieRoles

Resources

- ✧ Standalone Resources
  - Movies – can exist without Actors or MovieRoles
  - Actors – can exist without Movies or MovieRoles
- ✧ Dependent Resources
  - MovieRole
    - Depends on Movies to exist
    - Related to Actor, but can exist if relationship is severed

Rails - Resources

- ✧ `rails g scaffold` command
  - build templated code for CRUD operations
  - Mongoid or ActiveModel – additional implementation
- ✧ `rails g model Movie title`
- ✧ `rails g model Actor name`
- ✧ `rails g model MovieRole character`

Model Classes

```ruby
class Movie
  include Mongoid::Document
  include Mongoid::Timestamps
  field :title, type: String

  embeds_many :roles, class_name: "MovieRole"
end

```

```ruby
class MovieRole
  include Mongoid::Document
  field :character, type: String

  embedded_in :movie
  belongs_to :actor
end
```

```ruby
class Actor
  include Mongoid::Document
  include Mongoid::Timestamps
  field :name, type: String

  def roles
    Movie.where(:"roles.actor_id"=>self.id).map
      {|m|m.roles.where(:actor_id=>self.id).first}
  end
end
```

Summary

- ✧ Generated Model Classes
- ✧ Used the ORM to add dependency and relationship details
- ✧ Perform basic CRUD on these resources
- ✧ Ability to add more features

What's Next?

- ✧ URIs