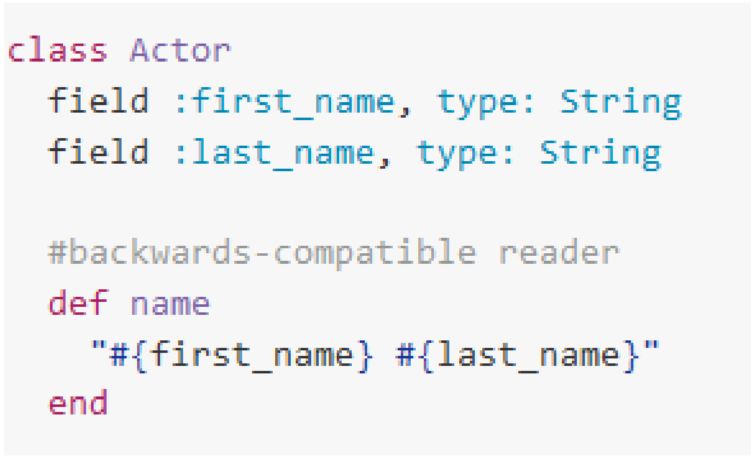
In this lecture, we will discuss…

- Versioning
- Vendor Media Type
- Backward compatible

Versioning

- Versioning – common practice in the industry
- `name` → `first_name` and `last_name`
- Backward compatible

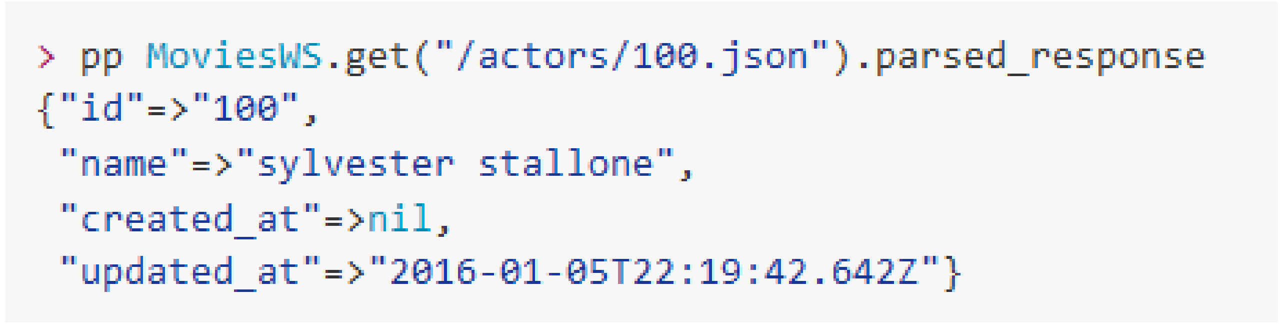


```
class Actor
  field :first_name, type: String
  field :last_name, type: String

  #backwards-compatible reader
  def name
    "#{first_name} #{last_name}"
  end
```

Usage

- ✧ Legacy Users can still use the service



```
> pp MoviesWS.get("/actors/100.json").parsed_response
{"id"=>"100",
 "name"=>"sylvester stallone",
 "created_at"=>nil,
 "updated_at"=>"2016-01-05T22:19:42.642Z"}
```

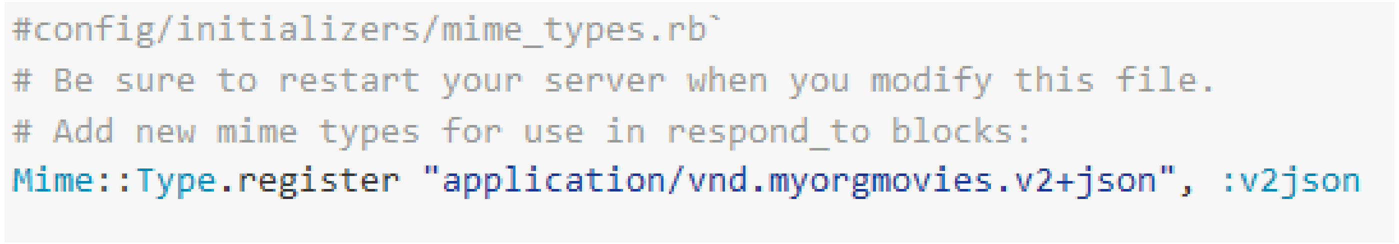
- ✧ Problem – Not able to get the new additions via JSON (without breaking the legacy users)

Solution - Versioning

✧ Vendor Media Type

- Define a different media type format

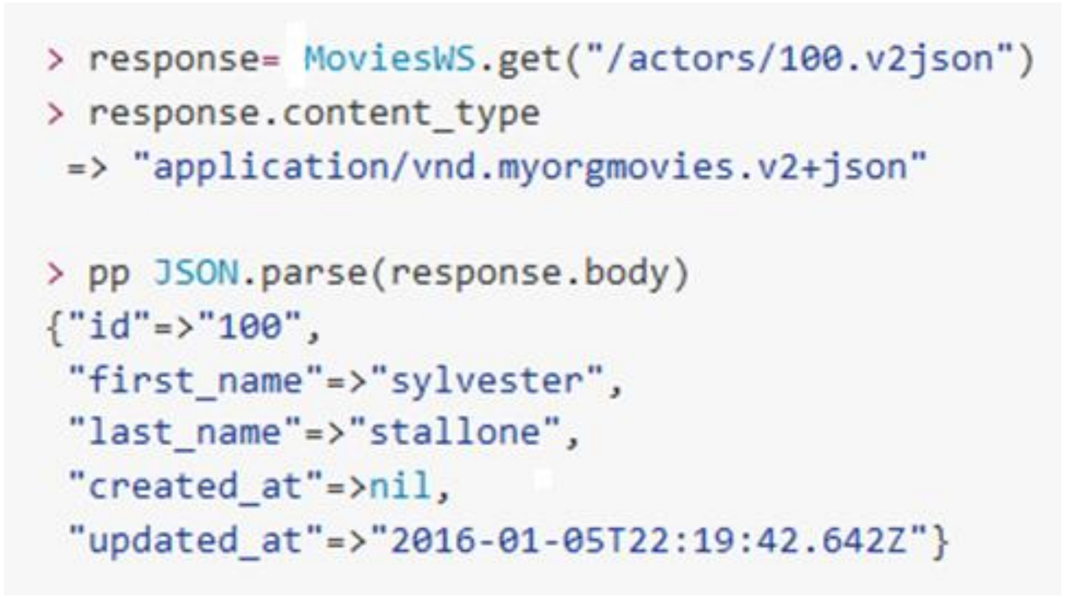
✧ Register the format (Ex: `v2json`)



```
#config/initializers/mime_types.rb`
# Be sure to restart your server when you modify this file.
# Add new mime types for use in respond_to blocks:
Mime::Type.register "application/vnd.myorgmovies.v2+json", :v2json
```

Demo – V2

✧ Access to v2 of the model

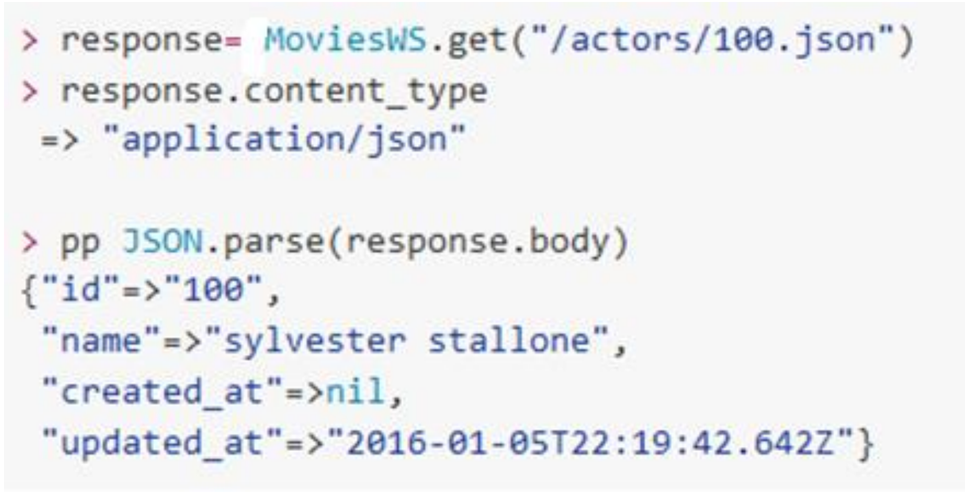


```
> response= MoviesWS.get("/actors/100.v2json")
> response.content_type
 => "application/vnd.myorgmovies.v2+json"

> pp JSON.parse(response.body)
{"id"=>"100",
 "first_name"=>"sylvester",
 "last_name"=>"stallone",
 "created_at"=>nil,
 "updated_at"=>"2016-01-05T22:19:42.642Z"}
```

Demo – V1

- ✧ Either define **`v1json`** from the start
- ✧ If not, default **`json`** as the "current" version



```
> response= MoviesWS.get("/actors/100.json")
> response.content_type
 => "application/json"

> pp JSON.parse(response.body)
{"id"=>"100",
 "name"=>"sylvester stallone",
 "created_at"=>nil,
 "updated_at"=>"2016-01-05T22:19:42.642Z"}
```

Summary

- ✧ Things are bound to change, consider versioning from the start

What’s Next?

- ✧ Content Negotiation