In this lecture, we will discuss…

- ✧ Default Rendering
- ✧ Controller Based Rendering
- ✧ Partials

Representations - Rendering

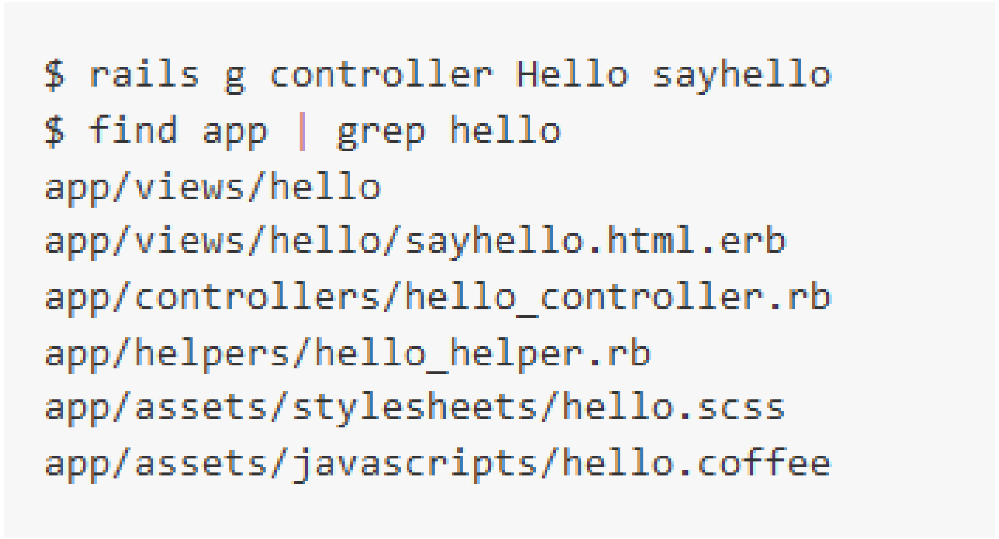
- ✧ Default Rendering
    - Controller derives data to render
    - View defaults to action requested
    - Content form based on client specification

Representations - Rendering

- Controller Based Rendering
  - Controller derives data to render
  - Controller makes decision on action to render
  - Controller makes decision on the format to render
    - Supply default format or overriding client specification

Example

✧ Create an **`Hello`** example with one **`hello`** action/view



```
$ rails g controller Hello sayhello
$ find app | grep hello
app/views/hello
app/views/hello/sayhello.html.erb
app/controllers/hello_controller.rb
app/helpers/hello_helper.rb
app/assets/stylesheets/hello.scss
app/assets/javascripts/hello.coffee
```

Example - changes

- ✧ hello_controller.rb

```ruby
#app/controllers/hello_controller.rb
class HelloController < ApplicationController
  def sayhello
  end
end
```

- ✧ routes.rb

```ruby
#config/routes.rb
  get 'hello/sayhello'
```

- ✧ rake routes

```
$ rake routes | grep hello
  hello_sayhello GET    /hello/sayhello(.:format)                  hello#sayhello
```

Default Rendering

- (action).(format).(handler suffix)
  - `sayhello.json.jbuilder` (JSON)
  - `sayhello.xml.builder` (XML)
  - `sayhello.text.raw` (RAW)

Default Rendering

- ✧ `#app/views/hello`
  - hello.json.builder → `json.msg @msg`
  - hello.xml.builder → `xml.msg @msg`
  - raw text message from: → `hello.text.raw`

Default Rendering - Demo

```
> response=MoviesWS.get("/hello/sayhello.json")
> response.header.content_type
 => "application/json"
> response.body
 => "{\"msg\":null}"
```

```
> response=MoviesWS.get("/hello/sayhello.xml")
> response.header.content_type
 => "application/xml"
> response.body
 => "<msg/>"
```

```
> response=MoviesWS.get("/hello/sayhello.txt")
> response.header.content_type
 => "text/plain"
> response.body
 => "raw text message from: app/views/hello/sayhello.text.raw\n"
```

Controller Derived Data

- Derive data from model
- Store in controller instance variable

```ruby
#app/controllers/hello_controller.rb
class HelloController < ApplicationController
  def sayhello
    @msg="hello world"
  end
end
```

```
> MoviesWS.get("/hello/sayhello.json").response.body
=> "{\"msg\":\"hello world\"}"

> MoviesWS.get("/hello/sayhello.xml").response.body
=> "<msg>hello world</msg>\n"
```

Controller Action Rendering

- ✧ Controller can have more active say in action rendered
- ✧ Define a route and action that takes a parameter

```
#config/routes.rb
  get 'hello/say/:something' => "hello#say"
```

- ✧ Controller action method with decision logic

```
#/hello/say/:something
def say
  case params[:something]
  when "hello" then @msg="saying hello"; render action: :sayhello
  when "goodbye" then @msg="saying goodbye"; render action: :saygoodbye
  when "badword" then render nothing: true
  else
    render plain: "what do you want me to say?"
  end
end
```

Supported View Format

```
#app/views/hello/saygoodbye.json.jbuilder
json.msg1 @msg
json.msg2 "so long!"
```

```
#app/views/hello/saygoodbye.xml.builder
xml.msg do
  xml.msg1 @msg
  xml.msg2 "so long!"
end
```

Demo

- Request `something=hello`
  - supplied by caller
- Resulting View: sayhello
  - determined by the controller
- Format `json` format
  - Specified by caller

- Request `something=goodbye`
  - supplied by caller
- Resulting View: goodbye
  - determined by the controller
- Format `xml` format
  - Specified by caller

Partials - Example

- Partials allow you to easily organize and reuse your view code in a Rails application
- Filename starts with an underscore
- Action-independent
- Default path: `_(partial).(format).(handler suffix)`
  - `app/views/actors/_actor.json.jbuilder`
  - `app/views/actors/_actor.xml.builder`

Summary

- ✧ Different rendering approaches

What’s Next?

- ✧ Versioning