Intro to MVC

MVC

What is MVC?

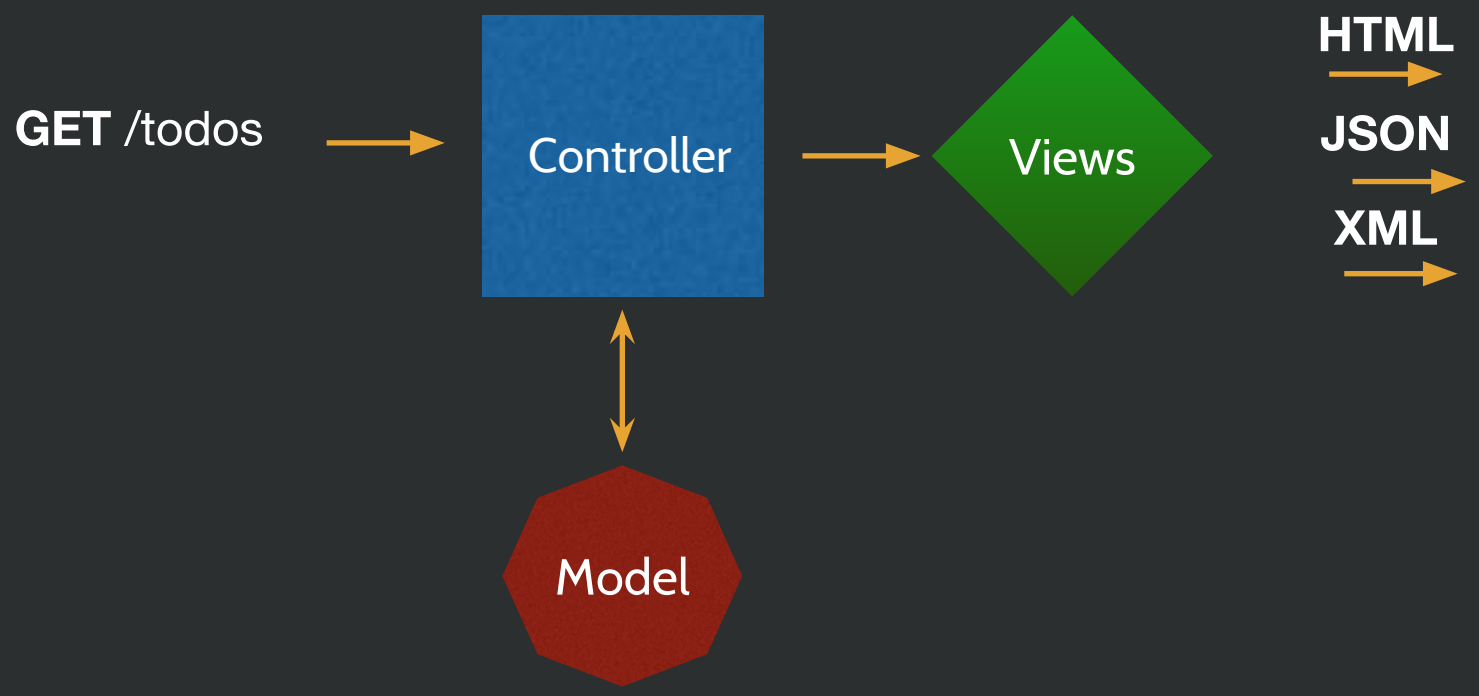
MVC

It stands for
M(odel)
V(iew)
C(ontroller)

MVC

It is a design pattern used in all types of application development

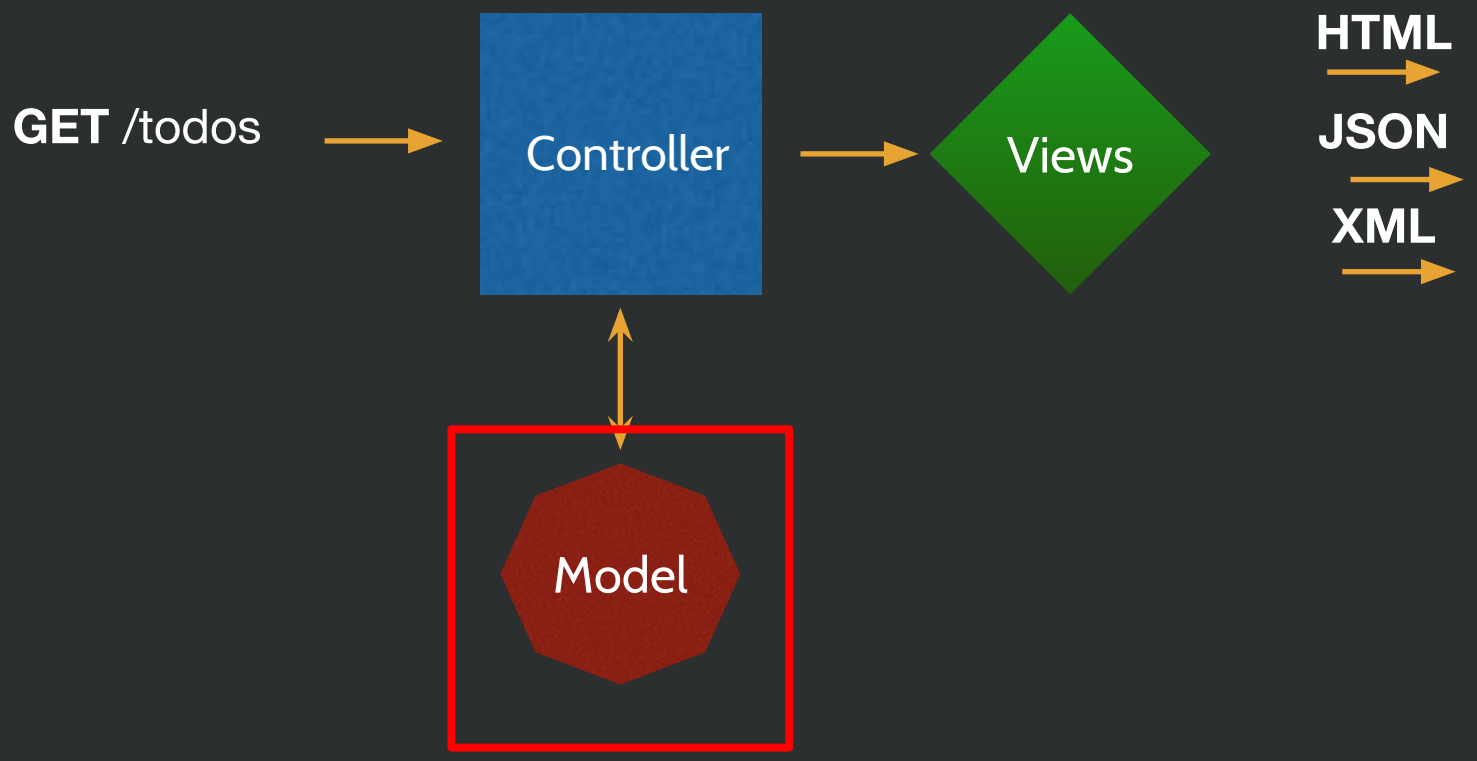
MVC



Models

Related to the Objects(classes) of your application

MVC



Models

A User class

Models

A Todo class

Models

A Shopping Cart

Models

Models should have most of the business logic of your application

IRON HACK

Models

IE: Calculating the total price of a shopping cart at checkout

Models

Authenticating a User

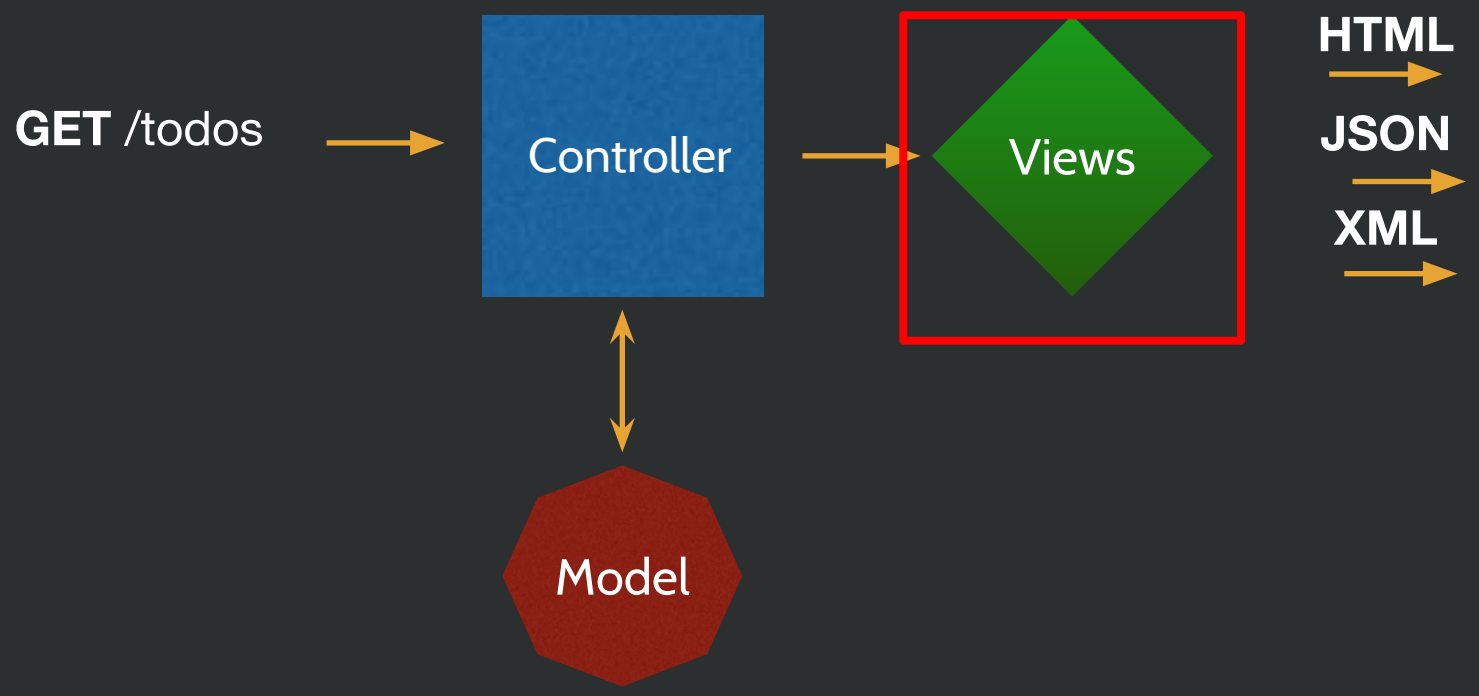
Models

Any of your methods that define how a class should act

IRON HACK

Views

MVC



Views

Views are related to your display

Views

They are responsible for taking data you give them, and presenting it

Views

Views should be dumb

IRON HACK

Views

There should be very little logic in your views

IRON HACK

Views

Logic being any kind of computation

Views

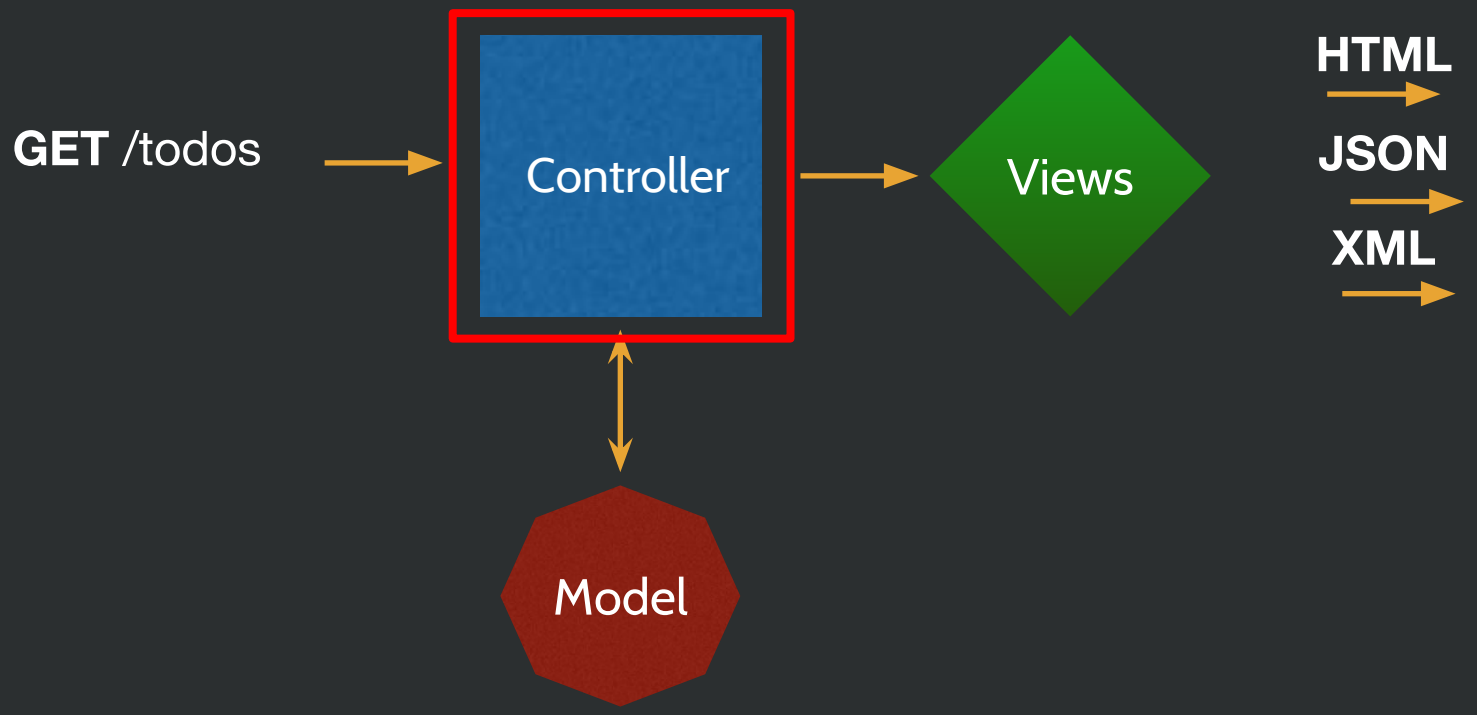
Calculating numbers, splitting an array, etc etc

Views

They simply present information in a nice way

Controllers

MVC



Controllers

Controllers are the coordinators in MVC.

Controllers

They pick the right data from the model

Controllers

And then they hand it off to the view

Controllers

Controllers should also be pretty dumb

IRON HACK

Controllers

They're just relaying information from the model to the view

Controllers

There shouldn't be much logic in the controller

IRON HACK

MVC

Why do we use MVC?

MVC

MVC is a pattern used to reinforce the separation of concerns.

MVC

Why would our Todo model worry about how the user views it?

MVC

Or worry about what route a user has to go to access a list of todos?

MVC

Just let the todo be a todo and do todo things

IRON HACK

MVC

Code Readability

#Bad

Let's check out my version of a Sinatra Todo without the MVC

IRON HACK

List of Todos

```
joshs_todos = [
                {content: "Buy Milk", time: 10},
                {content: "Walk the dog", time: 5}
              ]
get '/todos' do
  @todos = joshs_todos
  @hours_needed = @todos.reduce(0) do | sum, todo |
    sum + todo.time
  end
  html = ""
  html << "<h1>Welcome to the todo list</h1>"
  html << @todos.map do | todo |
    todo_html = ""
    todo_html << todo[:content]
    todo_html << todo[:time] + "<br>"
    todo_html
  end.join
  html << "You have: #{@hours_needed} of todos left"
end
```

#Bad

My Todo is a simple object

Adding a Todo: Form

```
get '/new_todo' do
  html = ""
  html << "<h1> Add a Todo to the Todo List </h1>"
  html << '
    <form action="/create_todo" method="post">
      <label for="content">
      <input type="text" id="content" name="content">
      <br>
      <label for="time">
      <input type="text" id="time" name="time">
      <br>
      <br>
      <input type="submit">
    </form>
  '
end
```

#Bad

Since Ruby returns the last item in my method, I have to add everything to this “html” variable

IRON HACK

Creating a Todo

```
post '/create_todo/:content/:time' do
  todo = {
    content: params[:content],
    time: params[:time]
  }
  joshs_todos.push(todo)
  redirect('/todos')
end
```

#Bad

So tell me what's going on here

#Bad

Exactly

#Bad

Let’s say I want to add a navigation bar to every page now

```ruby
get '/todos' do
  @todos = joshs_todos
  @hours_needed = @todos.reduce(0) do | sum, todo |
    sum + todo.time
  end
  html = ""
  html << '
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
  '
  html << "<h1>Welcome to the todo list</h1>"
  html << @todos.map do | todo |
    todo_html = ""
    todo_html << todo[:content]
    todo_html << todo[:time] + "<br>"
    todo_html
  end.join
  html << "You have: #{@hours_needed} of todos left"
end
```

```
get '/new_todo' do
  html = ""
  html << '
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
  '
  html << "<h1> Add a Todo to the Todo List </h1>"
  html << '
    <form action="/create_todo" method="post">
      <label for="content">
      <input type="text" id="content" name="content">
      <br>
      <label for="time">
      <input type="text" id="time" name="time">
      <br>
      <br>
      <input type="submit">
    </form>
  '
end
```

#Bad

My code is quickly starting to become messy

IRON HACK

#Bad

And there is very little on the page yet

The Refactoring

Let's abide by the MVC paradigm now

IRON HACK

The Refactoring

Let’s start with the models

IRON HACK

The Refactoring

Where do the models go?

The Refactoring

In the models folder, of course!

The Refactoring

```
├── app.rb
└── models
    ├── Todo.rb
    └── TodoList.rb
```

The Refactoring

```
require 'sinatra'
require 'sinatra/reloader'
require_relative('./models/Todo.rb')
require_relative('./models/TodoList.rb')
```

Models: TodoList

```
class TodoList
  attr_reader :list_of_todos
  def initialize
    @list_of_todos = []
  end

  def add_todo(todo)
    @list_of_todos.push(todo)
  end

  def total_time
    @list_of_todos.reduce(0) do | sum, todo |
      sum + todo.time
    end
  end
end
```

Models: Todo

```
class Todo
  attr_accessor :content, :time
  def initialize(content, time)
    @content = content
    @time = time
  end
end
```

The Refactoring

Then the controllers

Controller: Todos

```
get '/todos' do
  @todos = joshs_todos
  @hours_needed = @todos.total_time
  html = ""
  html << '
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
  '
  html << "<h1>Welcome to the todo list</h1>"
  html << @todos.map do | todo |
    todo_html = ""
    todo_html << todo[:content]
    todo_html << todo[:time] + "<br>"
    todo_html
  end.join
  html << "You have: #{@hours_needed} of todos left"
end
```

Controller: create_todos

```
post '/create_todo/:content/:time' do
  todo = Todo.new(params[:content], params[:time])
  joshs_todos.add_todo(todo)
  redirect('/todos')
end
```

Views

Our app is still a mess

Views

We still need to move all of our HTML code into a view

Views

Luckily, we already know where the views go

Views

As long as we have a folder named “views” sinatra will be smart, and find it

Views

```
get '/todos' do
  @todos = joshs_todos
  @hours_needed = @todos.total_time
  erb :todos
end
```

```
<!DOCTYPE html>
# todos.erb
<html>
  <head>
    <meta charset="utf-8">
    <title>Todos</title>
  </head>
  <body>
    <ul class="navbar">
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
    <% @todos.list_of_todos.each do | todo | %>
      <%= todo.content %>
      <br>
      <%= todo.time %>
    <% end %>
    <br>
    You have <%= @hours_needed %> hours of todos left.
  </body>
</html>
```

Views

Much better

Views

```
get '/new_todo' do
  erb :new_todo
end
```

```
<!DOCTYPE html>
# new_todo.erb
<html>
  <head>
    <meta charset="utf-8">
    <title>Todos</title>
  </head>
  <body>
    <ul class="navbar">
      <li><a href="#home">Home</a></li>
      <li><a href="#news">News</a></li>
      <li><a href="#contact">Contact</a></li>
      <li><a href="#about">About</a></li>
    </ul>
    <form action="/create_todo" method="post">
      <label for="content">
      <input type="text" id="content" name="content">
      <br>
      <label for="time">
      <input type="text" id="time" name="time">
      <br>
      <br>
      <input type="submit">
    </form>
  </body>
</html>
```

The Refactoring

When we break our code into pieces, it's easier to understand.

MVC

Most importantly, it's easy to understand what each piece does

IRON HACK

Pop Quiz

Where should each of these go?

Pop Quiz

Calculating the tax on a shopping cart

Pop Quiz

Finding 10 random store items out of a pre-defined list

Pop Quiz

Checking to see if a User's email is properly formatted

Pop Quiz

Displaying a list of Todos

Pop Quiz

Grabbing a list of 10 animals objects from our database

MVC

MVC is only one design pattern

MVC

There are many more, such as MVVM, MVP, and multi-tiered architecture

IRON HACK

MVC

There are also many different implementations of MVC

MVC

Sinatra style MVC is slightly different than Rails

IRON HACK

MVC

Rails MVC is different than MVC on the iOS

IRON HACK

MVC

While there are differences, many of the concepts are much the same

IRON HACK