

**Sinatra**

# What is it?

- It is a web application framework
- A Domain Specific Language
- A Ruby Gem

# What are those things?

## Web Application Framework

- A whole heap of code designed to help build web apps
- Meant to alleviate overhead with common activities
  - Accessing databases
  - Dealing with requests
  - etc.

# What are those things?

## Domain Specific Language

- A computer language targeted at a specific domain
  - By domain, we mean a specific set of functionality
- Some code that only tries to solve one problem, or has one purpose
- Sinatra gives Ruby the ability to respond to requests etc.

# What does Sinatra encourage?

- "Convention over configuration"
- "Principle of least surprise"

*Like most Ruby-related things*

# Important Links

- [Sinatra Website](#)
  - [Sinatra on Github](#)
  - [Sinatra Contrib on Github](#)

# Let's download it

```
gem install sinatra  
gem install sinatra-contrib
```

**Let's get into it!**



# The main bits of it

- Protocol
- Server name
- Subdomain
- Domain
- Port
- Path
- Parameters
- (Fragment or Anchor)

# The main parts of it

```
https://www.work.google.com:80/calendar?type=personal#home
```

# The main parts of it

```
https://www.work.google.com:80/calendar?type=personal#home
```

- https -> protocol
- www -> server name
- work -> subdomain
- google.com -> domain
- :80 -> port
- /calendar -> path
- ?type=personal -> parameters
- #home -> fragment or anchor

**We only care about the  
path for the moment!**

# How does Sinatra work?

We respond to requests (paths), that's how all web servers work

This is called routing. Routes are always matched in the order they are defined - first route it matches will be used

# How does that look?

```
get "/somePath" do  
  # Do something in here  
end
```

This is called literal routing:

- An HTTP method paired with a path

# Dynamic Routing

```
get '/hello/:name' do  
  # params[:name]  
end
```

This is called dynamic routing because it is:

- An HTTP method paired with a URL matching pattern

**But it isn't HTML just yet...**



# Views / Templates

- This uses something called ERB (Embedded Ruby) - it is very smart! But it requires a very particular setup
- We need a views folder, this is where all of our HTML will be stored (as ERB files)
- The first thing we always do is create a layout.erb file in the views folder - this will always be used
  - Any content you want on every page should be in here

# A layout.erb example

```
<!DOCTYPE html>
<html>
<head>
  <title>Our Basic Sinatra App</title>
</head>
<body>
  <nav>
    <a href="/">Home</a>
  </nav>

  <h1>Hello World</h1>

</body>
</html>
```

# Imagine a blog...

- You might want a footer and a navigation bar on every page
- But you also might want to show an actual blog post
- ERB is very good at this!

# ERB

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

This ERB line says:

*// Go find the post.erb file in the views folder*

But we need to have a place to put this content...

# Yield

```
<!DOCTYPE html>
<html>
<head>
  <title>Our Basic Sinatra App</title>
</head>
<body>
  <nav>
    <a href="/">Home</a>
  </nav>

  <%= yield %>

</body>
</html>
```

# Yield

Any piece of content that the erb command finds, will be placed in where the yield statement is

It makes all of our code very reusable and clean!

# Convention over Configuration

- views folder
- views/layout.erb file
- public folder
- public/style.css file

# Here is **your homework**

For more information about Sinatra, [see here](#).