In this lecture, we will discuss…

- `show`
- `new` and `create`
- `edit` and `update`
- `destroy`
- `paging`

Show

```
#GET /zips/{id}
#GET /zips/{id}.json
  before_action :set_zip, only: [:show, :edit, :update, :destroy]
  def set_zip
    @zip = Zip.find(params[:id])
  end

  def show
  end
```

New and Create

```
#POST /zips/new
  def new
    @zip = Zip.new
  end

#POST /zips
  def create
    @zip = Zip.new(zip_params)

    respond_to do |format|
      if @zip.save
        format.html { redirect_to @zip, notice: 'Zip was successfully created.' }
        format.json { render :show, status: :created, location: @zip }
      else
        format.html { render :new }
        format.json { render json: @zip.errors, status: :unprocessable_entity }
      end
    end
  end
```

"New" returns an initial prototype to the form to start editing

"Create" accepts the results and creates new instance in the database

Edit and Update

```
http://localhost:3000/zips/00002/edit

#GET /zips/{id}
  before_action :set_zip, only: [:show, :edit, :update, :destroy]
  def set_zip
    @zip = Zip.find(params[:id])
  end

  def edit
  end

#PUT /zips/{id}
  def update
    respond_to do |format|
      if @zip.update(zip_params)
        format.html { redirect_to @zip, notice: 'Zip was successfully updated.' }
        format.json { render :show, status: :ok, location: @zip }
      else
        format.html { render :edit }
        format.json { render json: @zip.errors, status: :unprocessable_entity }
      end
    end
  end
```

"Edit" retrieved the instance from the database

"Update" found the instance in the database and applied the changes

Destroy

```ruby
#DELETE /zips/{id}
  def destroy
    @zip.destroy
    respond_to do |format|
      format.html { redirect_to zips_url, notice: 'Zip was successfully destroyed.' }
      format.json { head :no_content }
    end
  end
```

Paging

- `gem 'will_paginate', '~>`

```
<table>
...
  <tbody>
    <% @zips.each do |zip| %>
      <% zip=toZip(zip) %>
      <tr>
        <td><%= zip.id %></td>
...
      </tr>
    <% end %>
  </tbody>
</table>
<%= will_paginate @zips %>
```

"will_paginate" – adds page properties from the database

Paging (controller and model)

```ruby
def index
  #@zips = Zip.all
  @zips = Zip.paginate(:page => params[:page])
end
```

Controller passes the value to model

```ruby
def self.paginate(params)
  Rails.logger.debug("paginate(#{params})")
  page=(params[:page] ||= 1).to_i
  limit=(params[:per_page] ||= 30).to_i
  offset=(page-1)*limit

  #get the associated page of Zips -- eagerly convert doc to Zip
  zips=[]
  all({}, {}, offset, limit).each do |doc|
    zips << Zip.new(doc)
  end

  #get a count of all documents in the collection
  total=all({}, {}, 0, 1).count

  WillPaginate::Collection.create(page, limit, total) do |pager|
    pager.replace(zips)
  end
end
```

Will translate the will_paginate input to all() query inputs

Will translate document array results to will_paginate result

Summary

- MVC – proven model

What's Next?

- Module 2