1. CRUD
2. RESTFUL ROUTES
3. SINATRA PREGUNTAS

CRUD

What does CRUD stand for?

CREATE
READ
UPDATE
DELETE

CRUD ---- HTTP VERB

```
CREATE ---- POST
READ ------ GET
UPDATE ---- PUT/PATCH
DELETE ---- DELETE
```

ActiveRecord - Create

Ways to create resources:

```
post '/caballos' do
  penelope = Caballo.create(params[:caballo])
  # -> <Caballo id=1 name=Penelope>
end
```

be careful - using the create method will always return the object regardless if it actually saves to the database (aka if it fails your validations)!

ActiveRecord - Create

Ways to create resources:

```ruby
post '/caballos' do
  penelope = Caballo.new(params[:caballo])
  # -> <Caballo id=nil name=Penelope>
  penelope.save
  # -> true
end
```

ActiveRecord - Create

key takeaway -

use #new & #save

OR

use #create & #valid?

por que? to better control the flow of your application AND avoid your application breaking during user experience

ActiveRecord - Update

```
put "/caballos/:caballo_id" do
  # we are receiving params[:caballo_name] from a form

  penelope = Caballo.find(params[:caballo_id])
  # -> <Caballo id=1 name=Penelope>


  penelope.name = params[:caballo_name]
  # -> "Priscilla"
  penelope.save
  # -> true
end
```

ACTIVERECORD - UPDATE

```
put "/caballos/:caballo_id" do
  penelope = Caballo.find(params[:caballo_id])
  # -> <Caballo id=1 name=Penelope>


  penelope.update_attributes(name: "Priscilla")
  # -> <Caballo id=1 name=Priscilla>
end
```

ActiveRecord - Read

```
#find

get '/caballos/:caballo_id' do
  penelope = Caballo.find(params[:caballo_id])
  # -> <Caballo id=1 name=Penelope>
end

#find_by

get '/caballos/:caballo_name' do
  penelope = Caballo.find_by(name: params[:caballo_name])
  # -> <Caballo id=1 name=Penelope>
end
```

ActiveRecord - Read

```
#all

get '/caballos' do
  @all_caballos = Caballo.all
end

#where

get '/caballos/:caballo_id' do
  @caballo = Caballo.where(id: params[:caballo_id]).first
end
```

ActiveRecord - Delete

```
delete "/caballos/:caballo_id" do
  penelope = Caballo.find(params[:caballo_id])
  # -> <Caballo id=1 name=Penelope>

  penelope.destroy
  # -> <Caballo id=1 name=Penelope>
end
```

PERO, TENEMOS UN PROBLEMA GRANDE

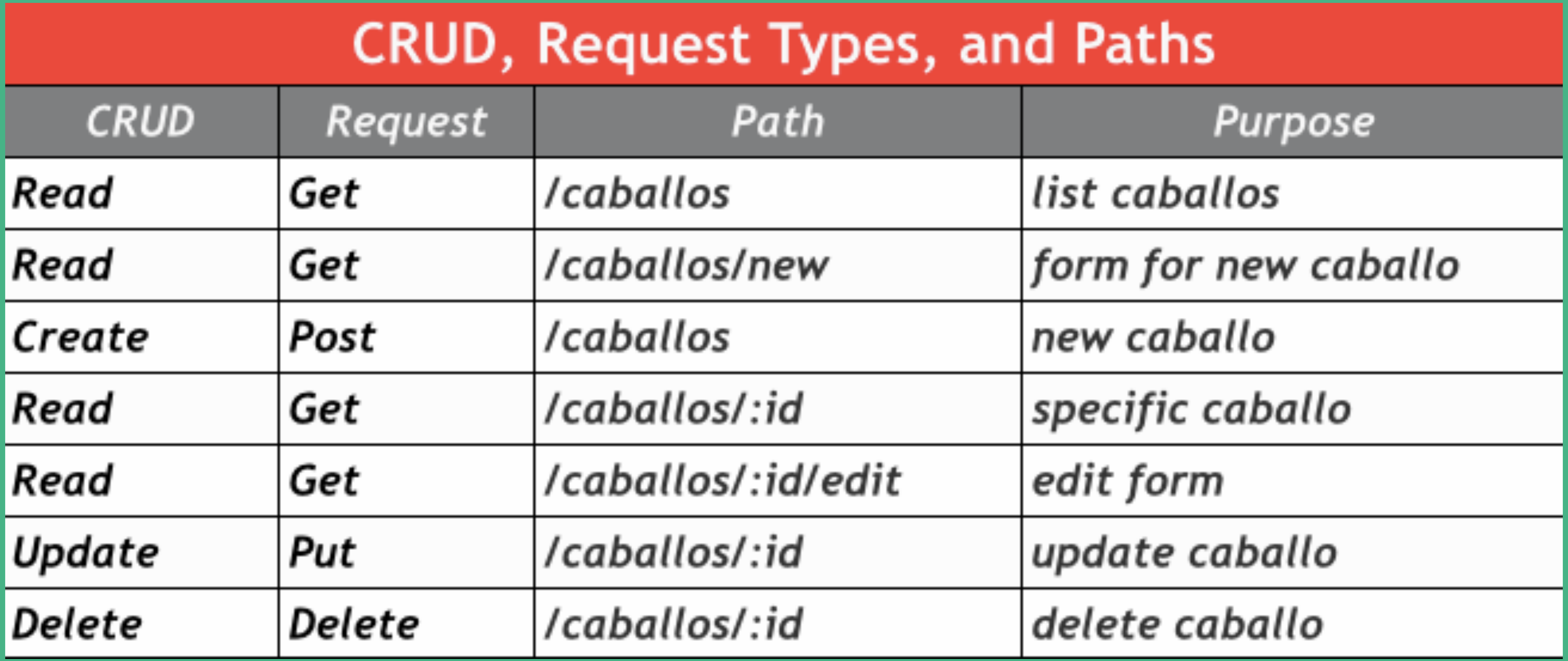
Modern day web browsers do not support `put` or `delete` HTTP requests

```
<form action="/caballos/<%= caballo.id %>" method="post">
  <input type=hidden name=_method value=put / >
  <input type='text' name='caballo_name' />
  <input type=submit value="Submit Now" />
</form>
```

here, a HTTP put request is made to /caballos/5

What is REST? What are RESTful routes?

RESTFUL CONVENTIONS



CRUD, Request Types, and Paths

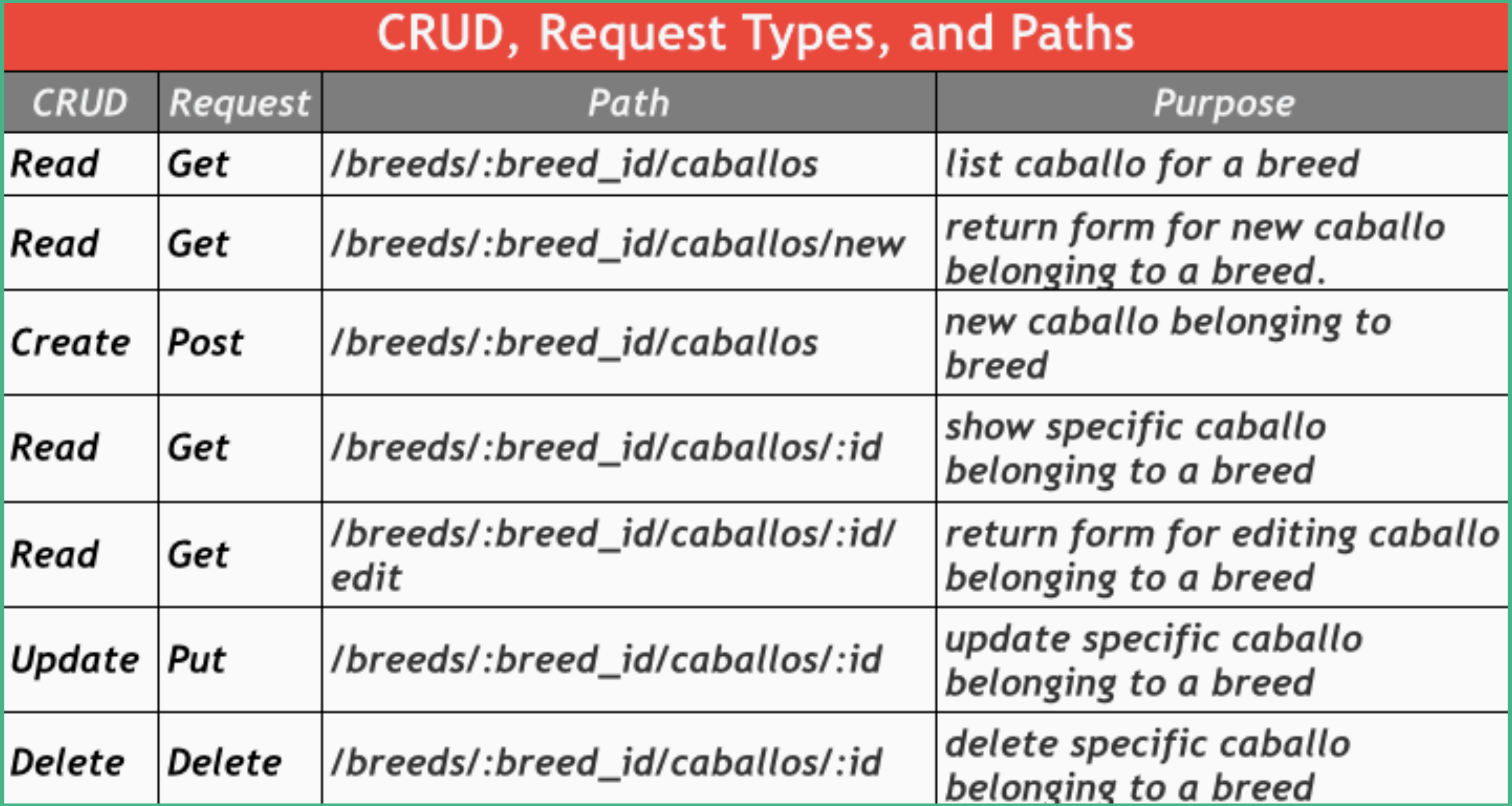
CRUD	Request	Path	Purpose
Read	Get	/caballos	list caballos
Read	Get	/caballos/new	form for new caballo
Create	Post	/caballos	new caballo
Read	Get	/caballos/:id	specific caballo
Read	Get	/caballos/:id/edit	edit form
Update	Put	/caballos/:id	update caballo
Delete	Delete	/caballos/:id	delete caballo

Why follow REST conventions?

Why follow RESTful conventions?

- Organization
- Best practices
- Clean, simple
- One less thing to think about

NESTED RESOURCES



CRUD, Request Types, and Paths

CRUD	Request	Path	Purpose
Read	Get	/breeds/:breed_id/caballos	list caballo for a breed
Read	Get	/breeds/:breed_id/caballos/new	return form for new caballo belonging to a breed.
Create	Post	/breeds/:breed_id/caballos	new caballo belonging to breed
Read	Get	/breeds/:breed_id/caballos/:id	show specific caballo belonging to a breed
Read	Get	/breeds/:breed_id/caballos/:id/ edit	return form for editing caballo belonging to a breed
Update	Put	/breeds/:breed_id/caballos/:id	update specific caballo belonging to a breed
Delete	Delete	/breeds/:breed_id/caballos/:id	delete specific caballo belonging to a breed

Rails Route Demo