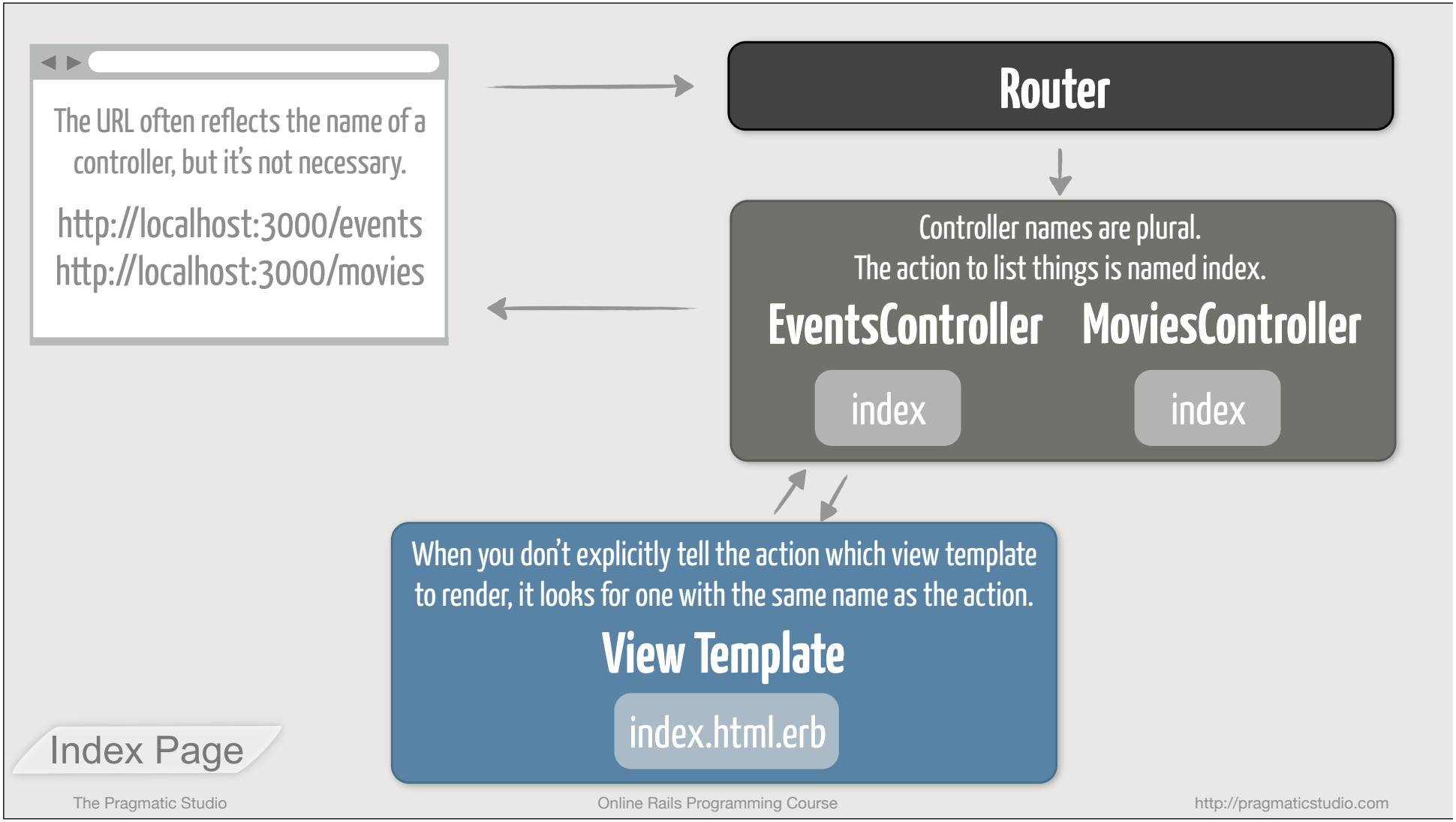
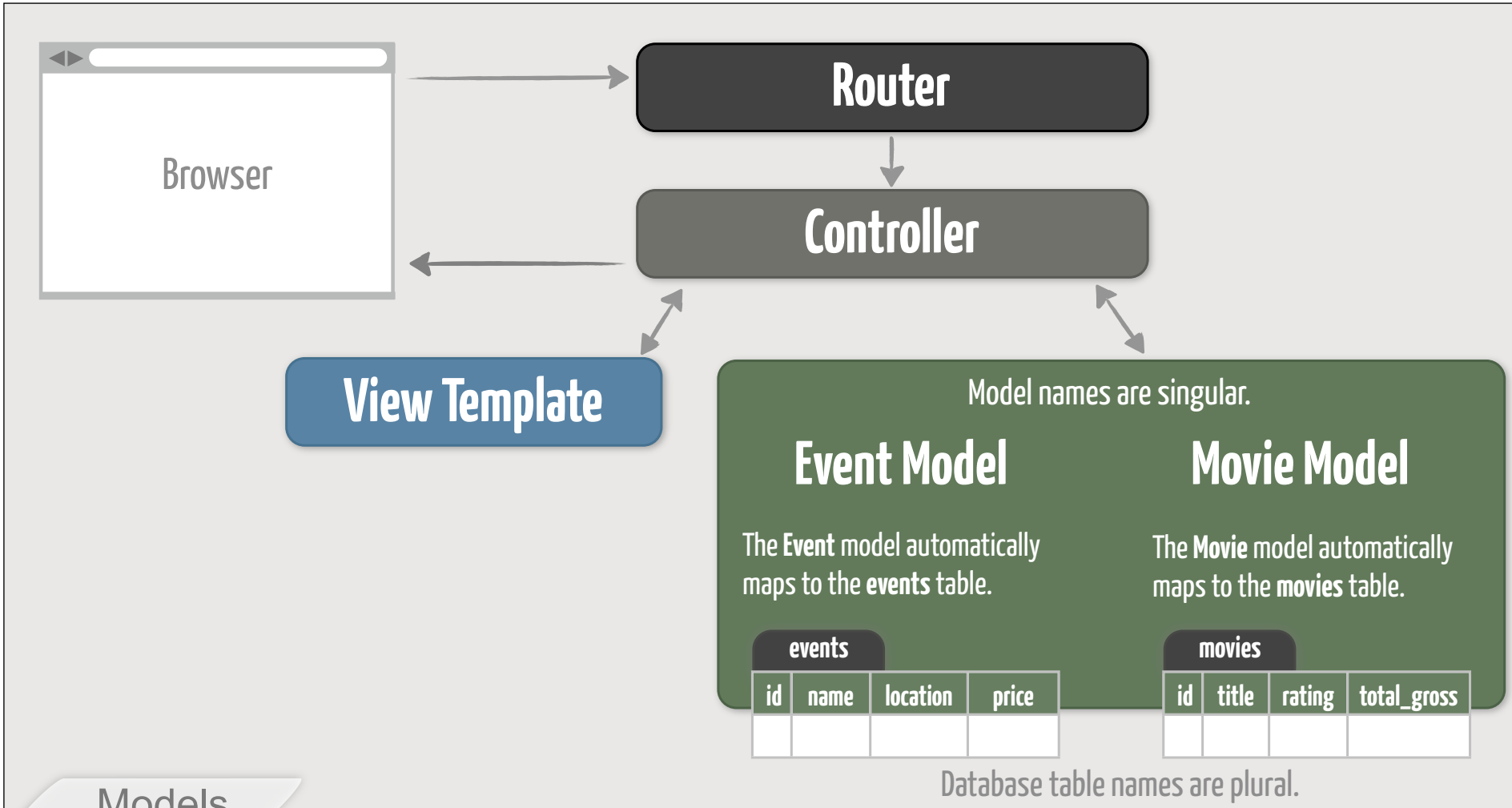
Rails Programming

CONVENTIONS

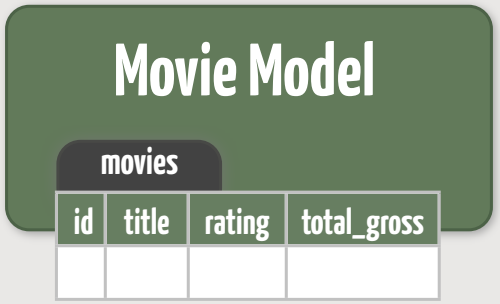
Copyright © 2013, The Pragmatic Studio. All Rights Reserved.
"Rails", "Ruby on Rails", and the Rails logo are registered trademarks of David Heinemeier Hansson. All rights reserved.


The URL often reflects the name of a controller, but it's not necessary.
http://localhost:3000/events
http://localhost:3000/movies
Router
Controller names are plural.
The action to list things is named index.
EventsController
index
MoviesController
index
When you don't explicitly tell the action which view template to render, it looks for one with the same name as the action.
View Template
index.html.erb

Index Page


Browser
Router
Controller
View Template
Model names are singular.
Event Model
Movie Model
The Event model automatically maps to the events table.
The Movie model automatically maps to the movies table.
events
id
name
location
price
movies
id
title
rating
total_gross
Database table names are plural.

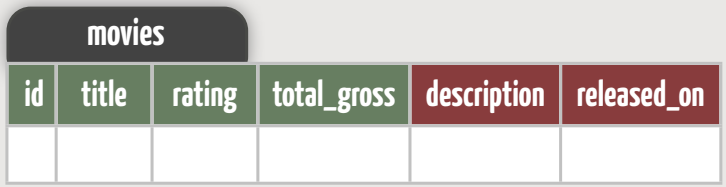
Models



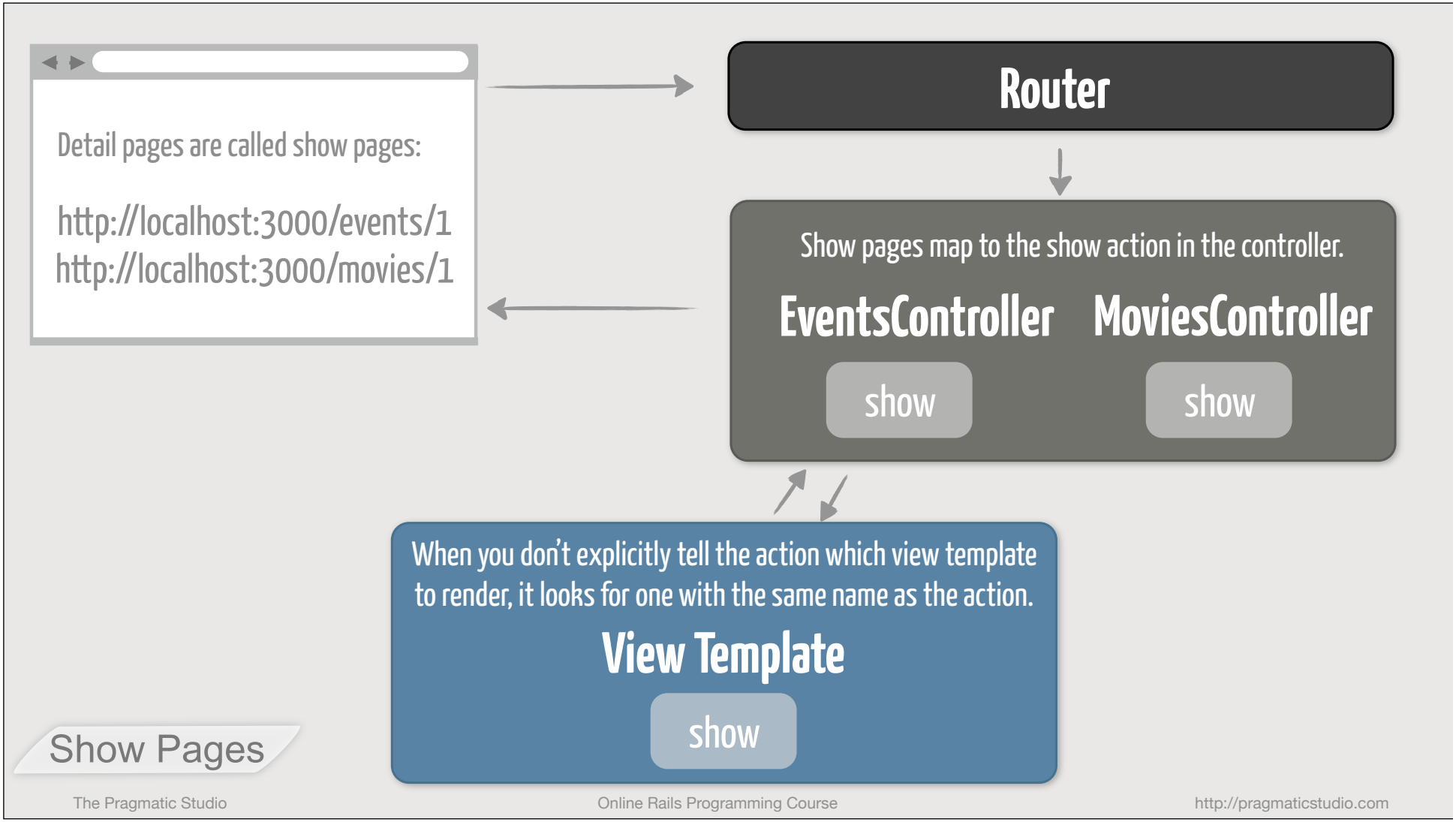
Name your migration using the format **AddXXXToYYY**, and Rails will assume you want to add the listed column names and types to the YYY table. (The XXX part is arbitrary.)

```
rails g migration AddFieldsToEvents description:text starts_at:datetime
```

```
rails g migration AddFieldsToMovies description:text released_on:date
```

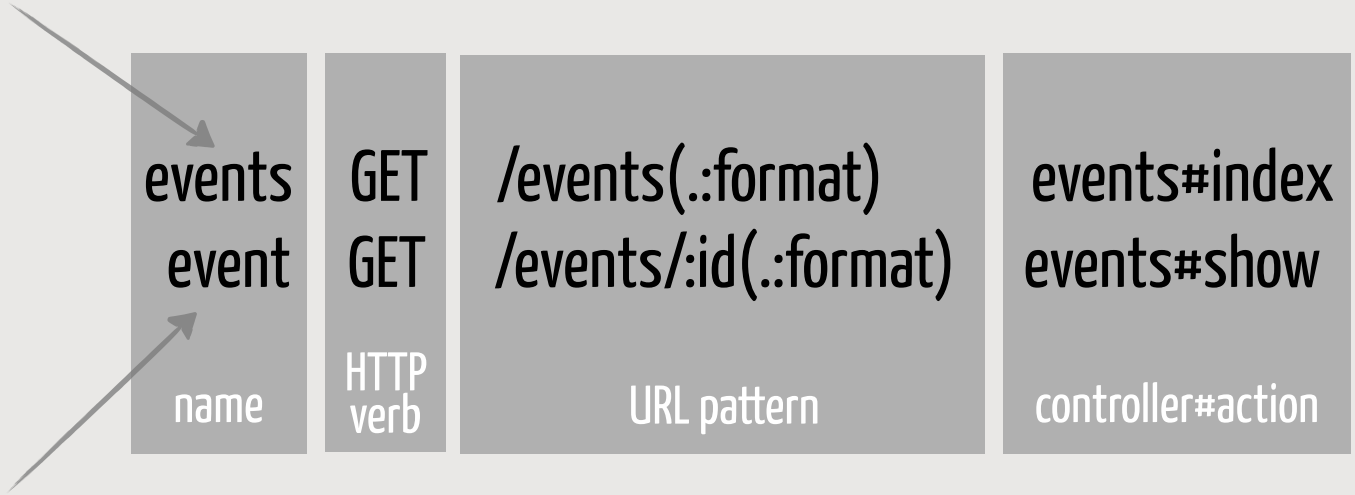


Migration Names


Detail pages are called show pages:
http://localhost:3000/events/1
http://localhost:3000/movies/1
Router
Show pages map to the show action in the controller.
EventsController
show
MoviesController
show
When you don't explicitly tell the action which view template to render, it looks for one with the same name as the action.
View Template
show

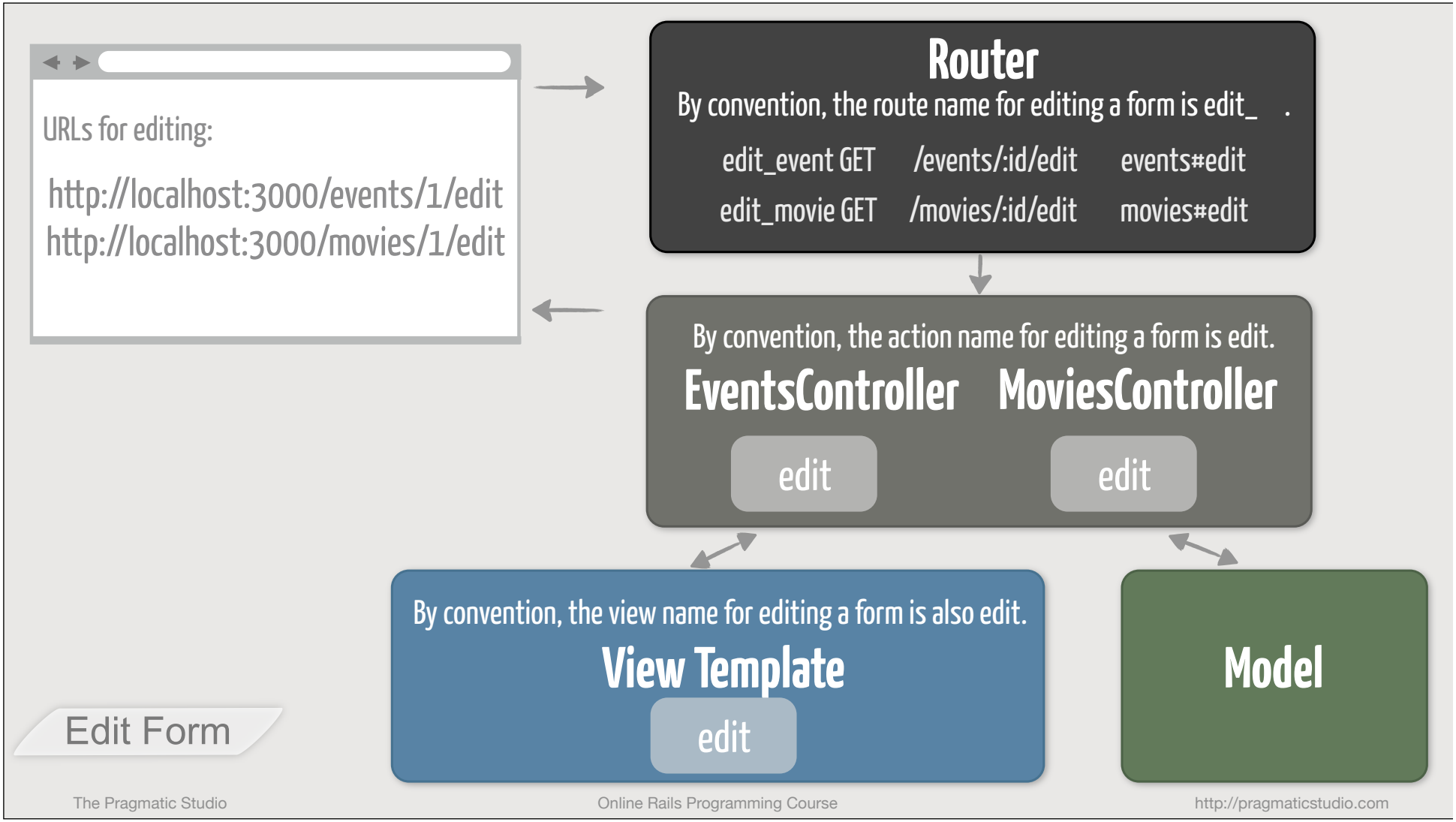
Show Pages

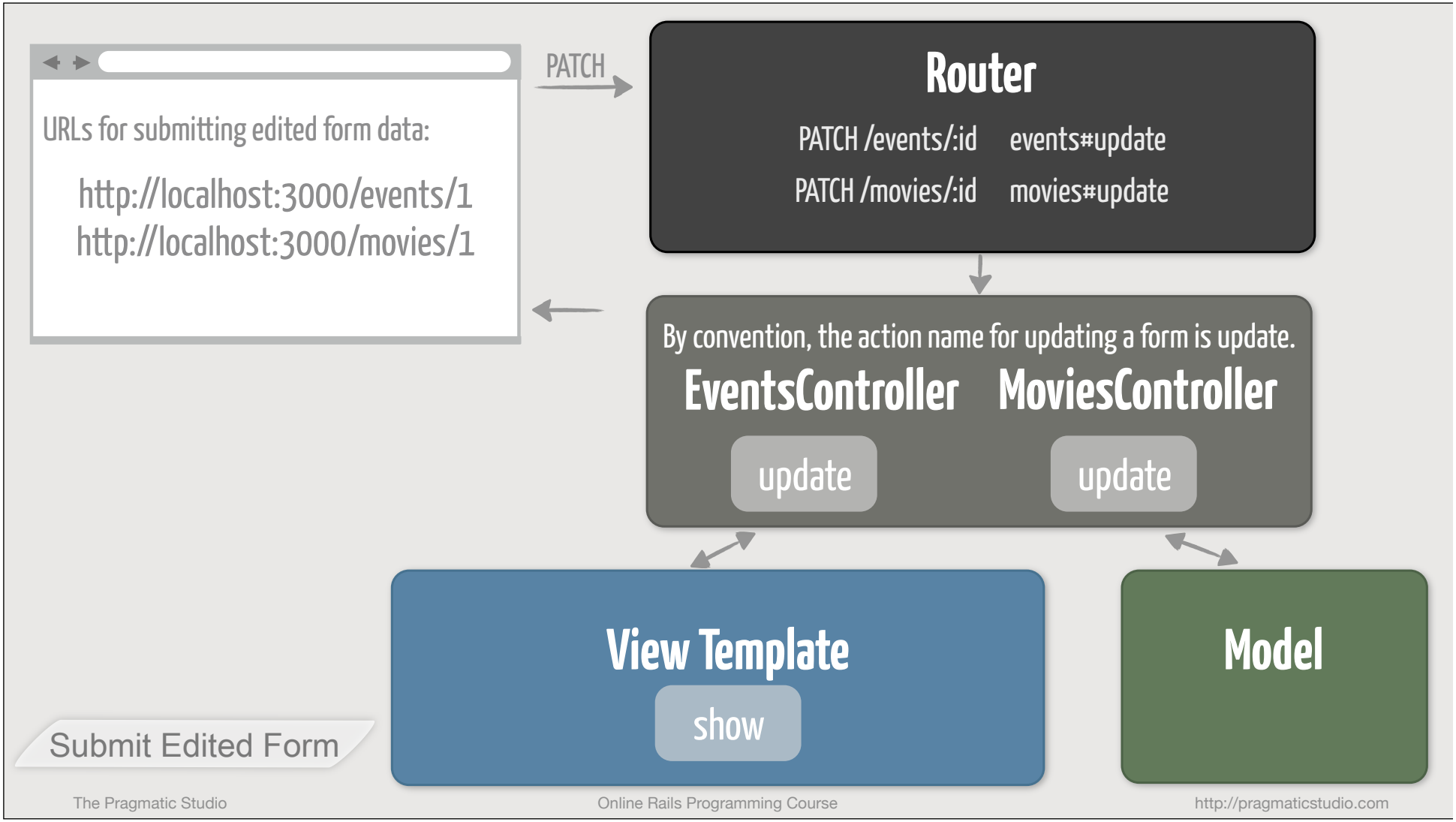
The name of the route that maps to the **index** action is **plural** because we're dealing with **multiple** things, such as multiple events or movies.

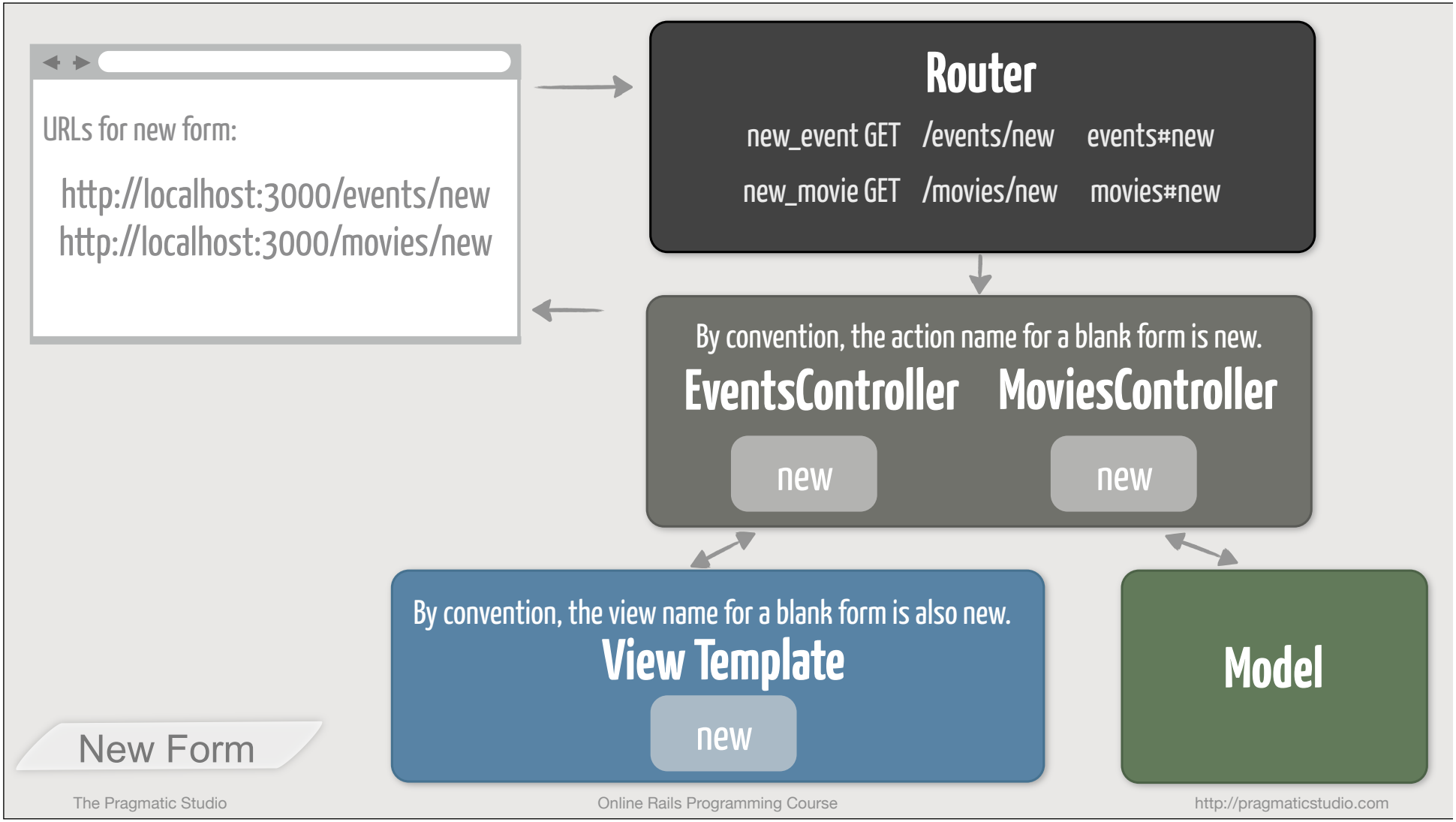


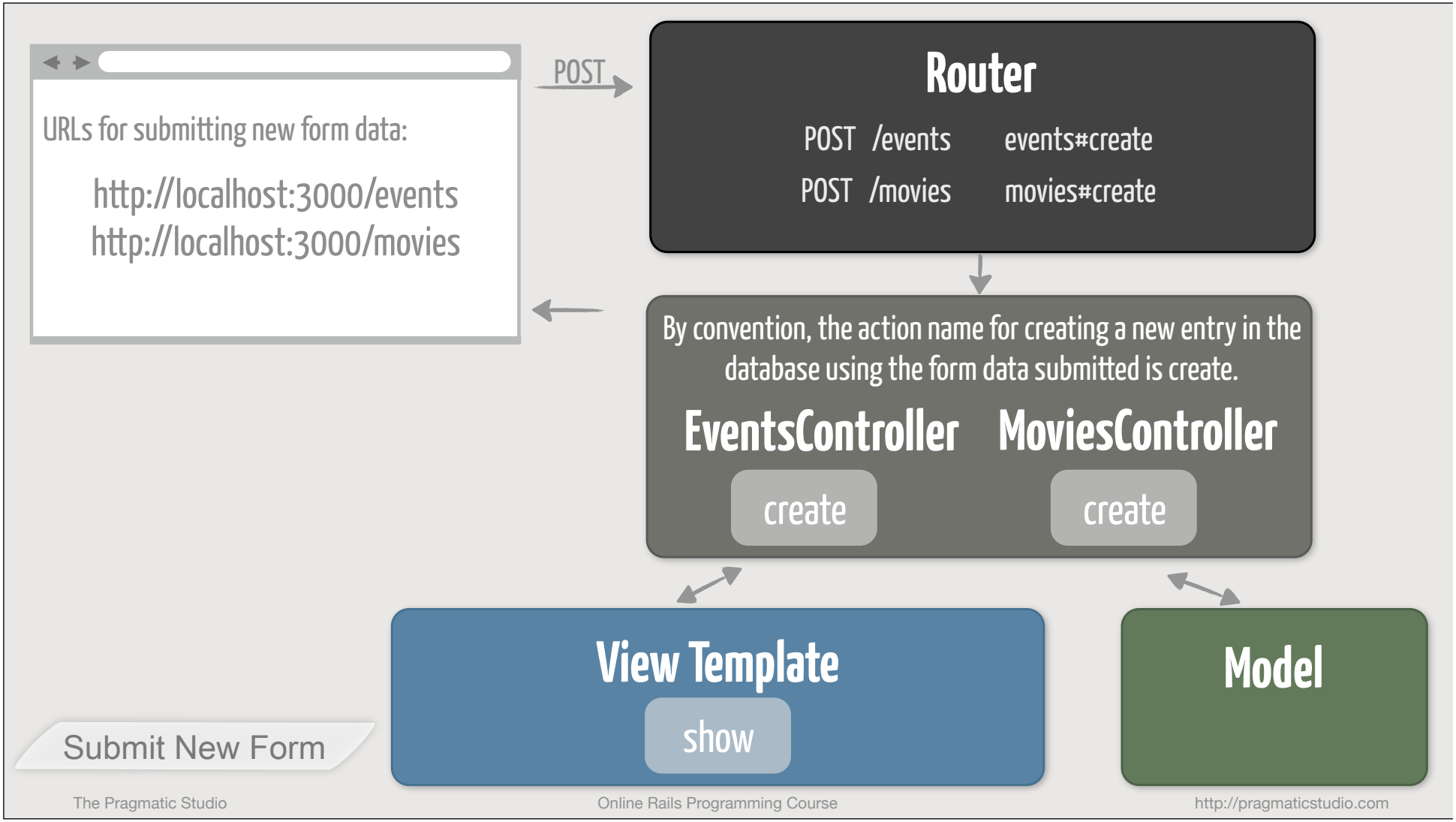
The name of the route that maps to the **show** action is **singular** because we're dealing with **one** thing, such as one event or movie.

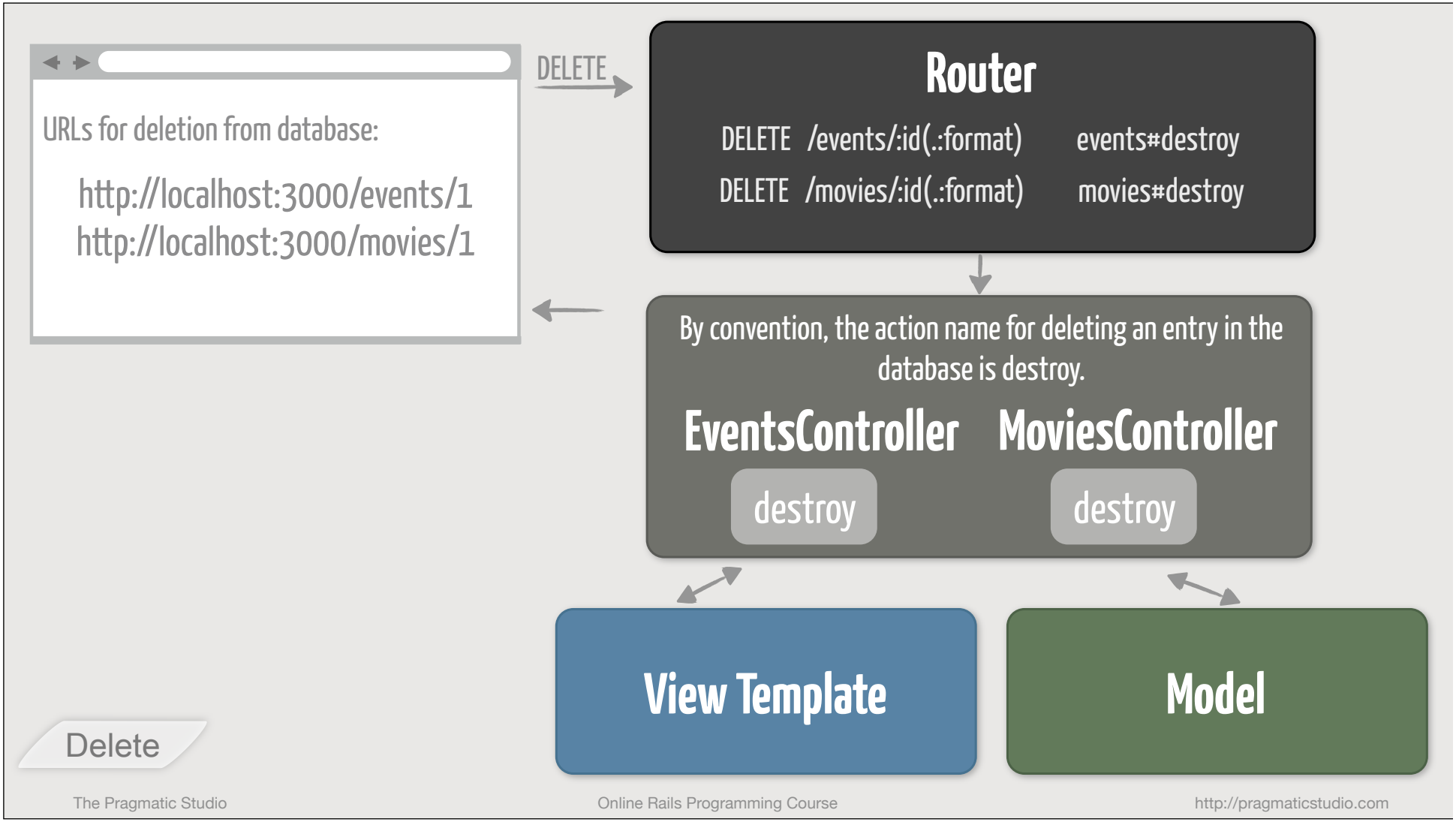
Route Names


URLs for editing:
http://localhost:3000/events/1/edit
http://localhost:3000/movies/1/edit
Router
By convention, the route name for editing a form is edit_ .
edit_event GET /events/:id/edit events#edit
edit_movie GET /movies/:id/edit movies#edit
By convention, the action name for editing a form is edit.
EventsController
edit
MoviesController
edit
By convention, the view name for editing a form is also edit.
View Template
edit
Model
Edit Form


URLs for submitting edited form data:
http://localhost:3000/events/1
http://localhost:3000/movies/1
PATCH
Router
PATCH /events/:id events#update
PATCH /movies/:id movies#update
By convention, the action name for updating a form is update.
EventsController
update
MoviesController
update
View Template
show
Model
Submit Edited Form


URLs for new form:
http://localhost:3000/events/new
http://localhost:3000/movies/new
Router
new_event GET /events/new events#new
new_movie GET /movies/new movies#new
By convention, the action name for a blank form is new.
EventsController
new
MoviesController
new
By convention, the view name for a blank form is also new.
View Template
new
Model
New Form

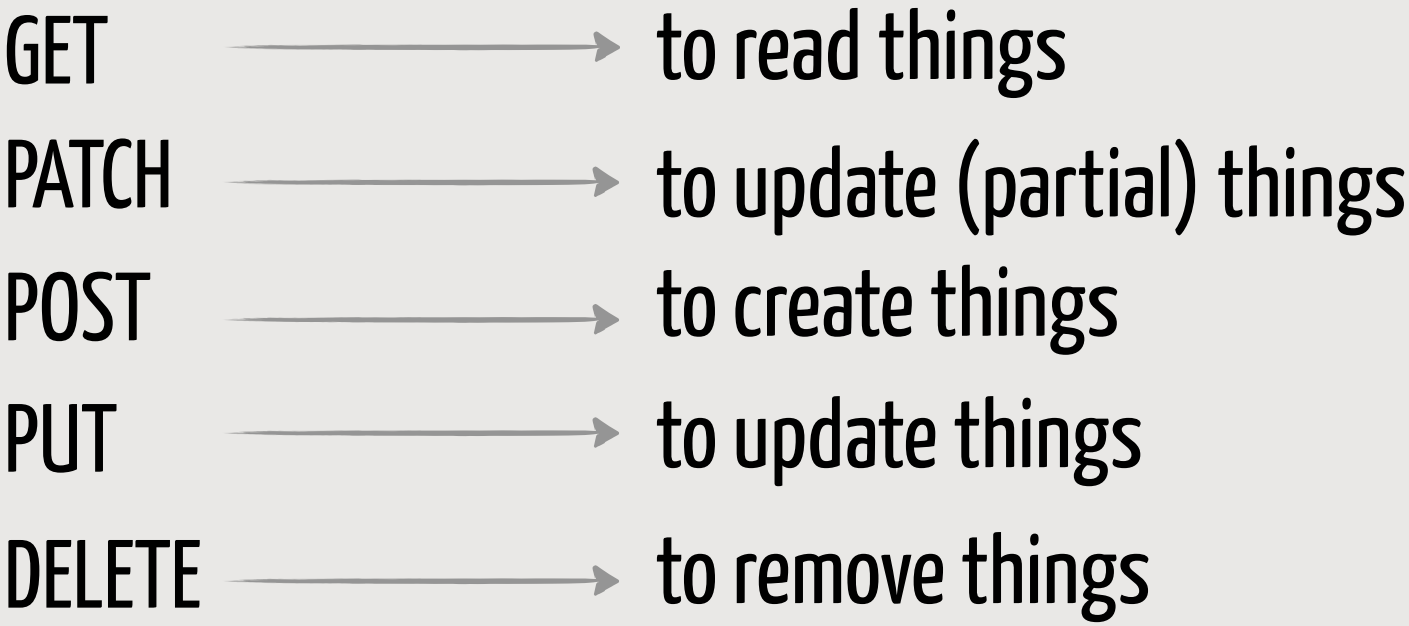

URLs for submitting new form data:
http://localhost:3000/events
http://localhost:3000/movies
POST
Router
POST /events events#create
POST /movies movies#create
By convention, the action name for creating a new entry in the database using the form data submitted is create.
EventsController
create
MoviesController
create
View Template
show
Model
Submit New Form


URLs for deletion from database:
http://localhost:3000/events/1
http://localhost:3000/movies/1
DELETE
Router
DELETE /events/:id(.:format) events#destroy
DELETE /movies/:id(.:format) movies#destroy
By convention, the action name for deleting an entry in the database is destroy.
EventsController
destroy
MoviesController
destroy
View Template
Model
Delete

resources :events

Name	HTTP Verb	URL Pattern	controller#action	
events	GET	/events	events#index	List All Events
event	GET	/events/:id	events#show	Show Specific Event
edit_event	GET	/events/:id/edit	events#edit	Show Edit Form
	PATCH	/events/:id	events#update	Update Event
	PUT	/events/:id	events#update	Update Event
	POST	/events	events#create	Create New Event
new_event	GET	/events/new	events#new	Show New Form
	DELETE	/events/:id	events#destroy	Delete Event

RESTful Conventions



Named Route Methods

name	HTTP verb	URL pattern	controller#action
events	GET	/events	events#index
event	GET	/events/:id	events#show

Helper Method	Generates
events_url	http://www.example.com/events
events_path	/events
event_url(id)	http://www.example.com /events/1
event_path(id)	/events/1

The **_path** method generates a **relative path**.

The **_url** method generates the **full URL**.

Event Model

```
class Event < ActiveRecord::Base
  has_many :registrations,
           :dependent => :destroy
end
```

events

id	name	location
1	BugSmash	Denver, CO
2	Hackathon	Austin, TX
3	Kata Camp	Dallas, TX

1 ——— *

Registration Model

```
class Registration < ActiveRecord::Base
  belongs_to :event
end
```

registrations

id	name	email	event_id
1	Larry	larry@stooges.com	1
2	Moe	moe@stooges.com	1
3	Moe	moe@stooges.com	3

The entity on the "many" (child) end needs a foreign key referencing the entity on the "one" (parent) end.

The name of this foreign key needs to be the name of the "one" (parent) entity, followed by "_id".

One-to-Many

Event Model

```
class Event < ActiveRecord::Base
  has_many :registrations,
           :dependent => :destroy
end
```

1 — *

Registration Model

```
class Registration < ActiveRecord::Base
  belongs_to :event
end
```

event.registrations

Returns an array of Registration objects that are associated with the event: all the registrations that have an event_id matching the event's ID.

registration.event

Returns the Event object associated with the registration: the event that matches the ID in the event_id column.

One-to-Many

Put as much **business logic** into the model as you can.

Event Model

```
def free?
  price.blank? || price.zero?
end

def spots_left
  capacity - registrations.count
end

def sold_out?
  spots_left.zero?
end
```

Movie Model

```
def flop?
 total_gross.blank? || total_gross < 50000000
end

def average_stars
  reviews.average(:stars)
end
```

Business Logic