


- ❖ Rails provides many **view helpers** so you can write HTML tags in Ruby!

```
<%= link_to "Google", "http://google.com" %>
<a href="http://google.com">Google</a>
```

```
<%= image_tag "starwars.jpg" %>

```

- ❖ There are also **URL helpers** that help you generate the paths to all the URLs in your application, so you don't have to remember them. Find out what routes are in your application using **rake routes**, then use the prefix to determine the URL helper, which is always the **prefix + _url**



	Prefix	Verb	URI Pattern	Controller#Action
	movies	GET	/movies(:format)	movies#index
		POST	/movies(:format)	movies#create
	new_movie	GET	/movies/new(:format)	movies#new
	edit_movie	GET	/movies/:id/edit(:format)	movies#edit
	movie	GET	/movies/:id(:format)	movies#show
		PATCH	/movies/:id(:format)	movies#update
		PUT	/movies/:id(:format)	movies#update
		DELETE	/movies/:id(:format)	movies#destroy
	root	GET	/	movies#index

Read this as:

The URL to my movies#index page (the list of all the movies) is **/movies**, and the URL helper is **movies_url**.

The URL to my movies#show page (a single movie's page) is **/movies/:id** (e.g. **/movies/123**), and the URL helper is **movie_url(123)**.

The URL to my movies#new page (the form to input a new movie) is **/movies/new**, and the URL helper is **new_movie_url**.

- ❖ There are also **form helpers** that help you write forms and their inputs without having to remember the crazy HTML.

```
<%= form_for @actor do |form| %>
  <%= form.label :name %>
  <%= form.text_field :name %>
  <%= form.button 'Create Actor' %>
<% end %>
```

```
<form action="/movies" method="POST">
  <label for="name">Name</label>
  <input type="text" id="name">
  <button type="submit">Create Movie</
button>
</form>
```

With Rails helpers

HTML equivalent

```
<%= form_for @actor do |form| %>
  <div class="form-group">
    <%= form.label :name %>
    <%= form.text_field :name, class: "form-control" %>
  </div>
  <%= form.button 'Create Actor', class: "btn btn-success" %>
<% end %>
```

With Bootstrap

- ❖ Knowing what you know already, if I asked you for the name of the studio that produced the movie with ID 3, you'd do this:

```
@movie = Movie.find_by(id: 3)
@studio = Studio.find_by(id: @movie.studio_id)
@studio.name
=> "Paramount"
```

- ❖ This works well enough, but as it turns out, there's an even better way! Rails supports a DSL (domain-specific language) for associating models. Wouldn't it be nice if we could do:

```
@movie = Movie.find_by(id: 3)
@movie.studio.name
=> "Paramount"
```

- ❖ We can! By keeping our domain model exactly the way it is, but add just a little bit of DSL code to our models:

```
class Movie < ActiveRecord::Base
  belongs_to :studio
end
```

```
class Studio < ActiveRecord::Base
  has_many :movies
end
```

- ❖ Many-to-many associations are common
- ❖ They can always be modeled as 2 one-to-many associations
- ❖ The trick is to identify the real-world “join model” in the middle
- ❖ You can use `has_many :through` to have Rails “connect the dots” for you

```
class Movie < ActiveRecord::Base

  has_many :roles
  has_many :actors, through: :roles

end
```

```
class Role < ActiveRecord::Base

  belongs_to :movie
  belongs_to :actor

end
```



This is the “join model”

```
class Actor < ActiveRecord::Base

  has_many :roles
  has_many :movies, through: :roles

end
```