Lesson 9:

Introducing models and ActiveRecord

Params

Rails stores browser-submitted values in an automatically-generated hash named `params`

The value from an HTML form field named review_name would be accessed as `params[:review_name]`

```
class ReviewsController < ApplicationController
    def index
        @review_name = params[:review_name]
    end
end
```

Remember that values in a controller's instance variables are accessible by its related layout and view templates

How do you create a simple data input form for your model?

Rails supports helper methods which enable views to quickly generate and manage HTML forms

A very simple form in a view intended to gather and display movie names could look like this

```
<h2>Recent book: <%= @movie_name %></h2>
<%= form_tag :action => 'index' do %>
    <p>
        Movie name:
        <%= text_field_tag 'movie_name', @movie_name %>
    </p>
    <%= submit_tag 'Add Movie' %>
<% end %>
```

Exercise:

Creating a simple input and display form

What is a model?

A model represents the data of the application and the rules to manipulate that data.

In Rails, models are primarily used for managing the rules of interaction with a corresponding database table. The application model is implemented using a software design pattern called Active Record.

What is a model?

Active Record represents rows of data as objects

All database interaction is managed through ActiveRecord objects, no SQL is required

ActiveRecord is built into Rails:
http://ar.rubyonrails.org/
http://api.rubyonrails.org/classes/ActiveRecord/Base.html

What is a model?

To create a basic model use the model generator with the rails command:

```
rails generate model <model_name>
```

Rails expects a model to be named for a **singular data type** (e.g., book, movie, user, etc.)

Rails generates code to:
– create a database table named plurally for this type (e.g., books, movies, users, etc.)
– create a sub-class of ActiveRecord::Base named with the model (e.g., Book, Movie, User, etc.)

What is a migration?

Migrations are a convenient way for you to alter your database in a structured and organized manner. You could edit fragments of SQL by hand but you would then be responsible for telling other developers that they need to go and run them.

In Rails, a migration is a class derived from ActiveRecord::Migration which defines how a database table should be built. It is generated in the db/migrate directory when a basic model is generated.

What is a migration?

The change method defines columns to be created and their data types

```
class CreateReviews < ActiveRecord::Migration
    def change
        create_table :reviews do |t|
            t.string :movie_name
            t.string :reviewer_name
            t.string :synopsis
            t.timestamps
        end
    end
end
```

ActiveRecord and migrations are discussed in more detail later in the course

What types of data can ActiveRecord hold?

`:binary`	binary large objects (“BLOBS”) like pictures
`:boolean`	true|false values
`:date`	db specific representation for a date
`:datetime`	db specific representation for a date and time
`:decimal`	fractional numbers
`:float`	fractional numbers
`:integer`	non-fractional numbers
`:primary_key`	unique ID for each record in a table
`:string`	list of up to 255 characters
`:text`	list of characters up to db capacity for this type
`:time`	db specific representation for a time
`:timestamp`	db specific representation for a time

How do you manually edit a model?

A model is a class derived from ActiveRecord::Base which represents a single record of data for a table

It is generated – except for accessors related to columns – in the app/models directory when a basic model is generated

Accessors are defined for the data each record of the corresponding table will expose

```
class Review < ActiveRecord::Base
    attr_accessible :movie_name
    attr_accessible :reviewer_name
    attr_accessible :synopsis
end
```

Introducing Rake

What is Rake?

Rake is Rails's software task management tool, often used to automate moving, compiling, and deleting Ruby files

Rake executes tasks defined in "rakefiles" which describe tasks to be completed

Introduction to using rake
http://guides.rubyonrails.org/command_line.html#rake

User guide to rake
http://docs.rubyrake.org/user_guide/index.html

Rake is discussed in more detail throughout the course

What does rake db:migrate do?

Rake tasks may be grouped into namespaces for organization, like `rake db:[task_name]` for database related tasks

`rake db:migrate`
executes migrations defined in a Rails application and modifies its database tables

Rake tasks for databases are discussed in more detail in the later lesson specifically on migrations

How do you interact with a database using a model?

Models support methods to create, read, update, and delete records

A model can be used to create a new record with specified values, and return a true or false for success or failure

```
@review = Review.create({
    movie_name: ‘Tron’,
    reviewer_name: ‘Rik’,
    synopsis: ‘Light cycles, etc.’
})
```

How do you interact with a database using a model?

A model can retrieve all records, all records where certain values match, or find one record by an ID value:

```
@reviews = Review.all

@reviews = Review.where(reviewer_name: “Rik”)

@review = Review.find(2)
```

Models and their methods are discussed in more detail later in this course

How do you create a simple data display form for your model?

Rails supports helper methods which enable views to quickly generate and manage HTML display

A very simple form in a view intended to display a list movie names could look like this

```
<p>Movie list:</p>
<ul>
    <% @movies.each do |movie| %>
    <li><%= movie.movie_name %></li>
    <% end %>
</ul>
```

Forms, input, validation, and display are discussed in more detail later in the course

Debugging

Rails supports a debug function which can be displayed as a view template expression to display browser input

```
<%= debug params %>
```

Other arguments to the debug function, each providing different information sets, include assigns, controller, base_path, flash, request, response, session

Exercise:
Adding model to our site