# **Welcome to Rails**

Rails is a web-application framework that includes everything needed to create database-backed web applications according to the [Model-View-Controller (MVC)](http://en.wikipedia.org/wiki/Model-view-controller) pattern.

Understanding the MVC pattern is key to understanding Rails. MVC divides your application into three layers, each with a specific responsibility.

The *Model layer* represents your domain model (such as Account, Product, Person, Post, etc.) and encapsulates the business logic that is specific to your application. In Rails, database-backed model classes are derived from ActiveRecord::Base. Active Record allows you to present the data from database rows as objects and embellish these data objects with business logic methods. You can read more about Active Record in its [README](https://github.com/rails/rails/blob/master/activerecord/README.rdoc). Although most Rails models are backed by a database, models can also be ordinary Ruby classes, or Ruby classes that implement a set of interfaces as provided by the Active Model module. You can read more about Active Model in its [README](https://github.com/rails/rails/blob/master/activemodel/README.rdoc).

The *Controller layer* is responsible for handling incoming HTTP requests and providing a suitable response. Usually this means returning HTML, but Rails controllers can also generate XML, JSON, PDFs, mobile-specific views, and more. Controllers load and manipulate models, and render view templates in order to generate the appropriate HTTP response. In Rails, incoming requests are routed by Action Dispatch to an appropriate controller, and controller classes are derived from ActionController::Base. Action Dispatch and Action Controller are bundled together in Action Pack. You can read more about Action Pack in its [README](https://github.com/rails/rails/blob/master/actionpack/README.rdoc).

The *View layer* is composed of "templates" that are responsible for providing appropriate representations of your application's resources. Templates can come in a variety of formats, but most view templates are HTML with embedded Ruby code (ERB files). Views are typically rendered to generate a controller response, or to generate the body of an email. In Rails, View generation is handled by Action View. You can read more about Action View in its [README](https://github.com/rails/rails/blob/master/actionview/README.rdoc).

Active Record, Active Model, Action Pack, and Action View can each be used independently outside Rails. In addition to that, Rails also comes with Action Mailer ([README](https://github.com/rails/rails/blob/master/actionmailer/README.rdoc)), a library to generate and send emails; Active Job ([README](https://github.com/rails/rails/blob/master/activejob/README.md)), a framework for declaring jobs and making them run on a variety of queueing backends; Action Cable ([README](https://github.com/rails/rails/blob/master/actioncable/README.md)), a framework to integrate WebSockets with a Rails application; and Active Support ([README](https://github.com/rails/rails/blob/master/activesupport/README.rdoc)), a collection of utility classes and standard library extensions that are useful for Rails, and may also be used independently outside Rails.