Ruby on Rails Project Layout

Ruby on Rails lays out the code of a project by placing the layers of the application into separate directories. This document briefly describes the location and purpose of each layer.

App Directory

app/controllers
The controllers are the entry point for any request into your application. They are responsible for orchestrating the request and returning an appropriate response.

app/decorators
Decorators wrap a model with additional methods to help with the display of data. They are a method to unify common view logic in a single location.

app/helpers
Helpers allow you to share common reusable methods throughout your application.

app/jobs
Jobs let you declare a background job that can be run with a selection of different backends. This keeps your background tasks separated from the implementation at the systems level.

app/mailers
The entry point for sending an email from your application. They orchestrate the sending of the email by loading the appropriate data from the database and rendering the appropriate view.

app/models
Your models represent the abstract logic and data of your application. ActiveRecord models store their data within the configured database.

app/packs
Contains all of your frontend resources, such as JavaScript, CSS and images which are then compiled into a pack for serving to the web browser.

app/views
The views define the output of the response from a controller or mailer. Typically this will take the form of HTML being output via some form of templating language such as Haml.

Config Directory

The config directory contains a variety of configuration files for different parts of the application, including deployment, database and routes.

DB Directory

The db directory contains files that define the structure of your database. The migrate subdirectory contains a timestamped set of files which describe each step of transforming the database from the initial version of the database into the most up to date version. These

files are executed from the last version executed to the current version when **rails db:migrate** is run.

The seeds file is provided to help set some initial data in your applications database and can be run with the command **rails db:seed**.

Log Directory

This contains the output of both the server and the deployment task in capistrano.log.

Spec Directory

The spec directory contains the application's automated testing files.

spec/factories
Factories are defined with the FactoryBot gem. They let you share logic for creating records in the database within tests. These are referred to as fixtures.

spec/features
This directory contains your application's feature tests which drive a part of the application directly. They test the application "from the outside" checking that performing certain actions leads to the correct response from the application.

spec/models
This directory contains the model tests for your application which test a single class, usually from your model in isolation.

Storage Directory

This is the default location to store files uploaded with ActiveStorage. This directory is not publicly accessible but by default ActiveStorage files are publicly accessible at hard to guess URLs. You can restrict access files by implementing Authenticated Controllers.

Public Directory

This directory is publically accessible for your application. There is no need to manually place assets in here as most are better placed in the packs directory. Do not place files in this directory that need access control applying to them as they will be available to anyone accessing the web application.

Vendor Directory

The vendor directory contains code that has been brought into the project from a third-party. This can make it easier to bring in updates to code that has to be copied in. In general using gems or code that is maintained as part of the project is preferable to placing code in the vendor directory. Many projects will never make use of this directory.