Understanding Ruby on Rails

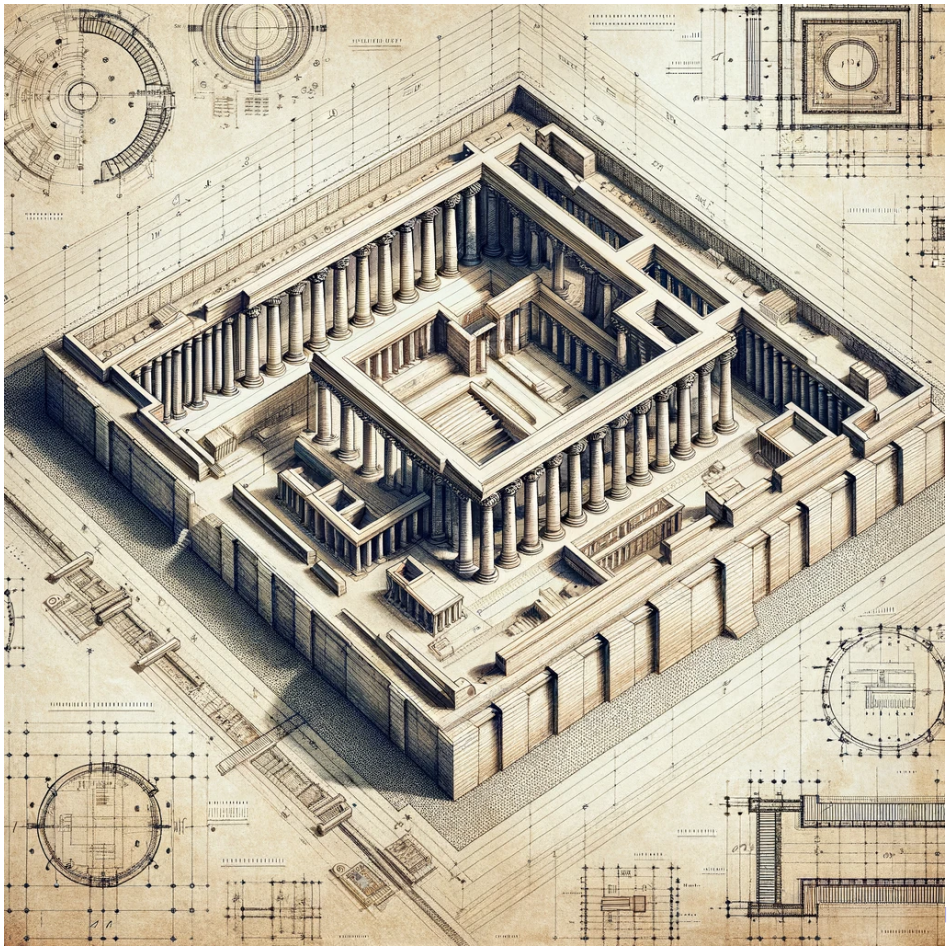
Through the Construction of Solomon's Temple

Bro. NJ Franklin

March 11, 2024

Foundation and Layout

Just like the detailed plan needed for Solomon's Temple, your Rails application begins with the `routes.rb` file. This serves as the blueprint, outlining all pathways within your app, guiding visitors (requests) to their destinations (controllers and actions).



Entrances

Solomon's Temple had specific entrances for different purposes. Similarly, Rails routes use HTTP verbs (GET, POST, PUT, DELETE) alongside paths to define how users interact with your application—whether they're viewing pages (GET) or submitting forms (POST).

Figure: Entrances analogy

Priests and Ceremonies

In the temple, priests performed ceremonies in designated areas. In Rails, controllers and actions take on this role, handling specific tasks like retrieving data or rendering views, based on the route that was followed.

Figure: Controllers and actions analogy

Dedications

The temple was divided into sections each with a unique purpose, from the Holy of Holies to the Courts. Rails routes can be dedicated to specific functionalities using ‘resources‘, which automatically creates a standard set of routes for objects like articles or profiles.

Figure: Resources analogy

Pathways

Named pathways in the temple guided people to their destinations. Rails offers named routes, such as `articles_path` or `new_article_path`, providing clear, readable paths to different parts of your application.

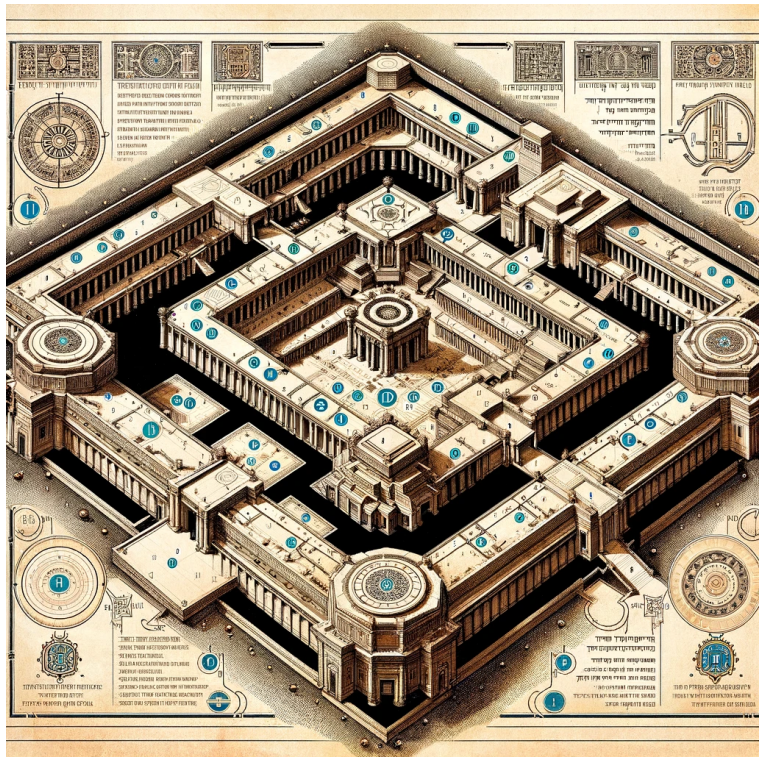


Figure: Named routes analogy

Views

The Views in Rails are like the windows and decorations of Solomon's Temple, shaping how visitors perceive the temple's interior. They are responsible for presenting the data to the user in a specific format.

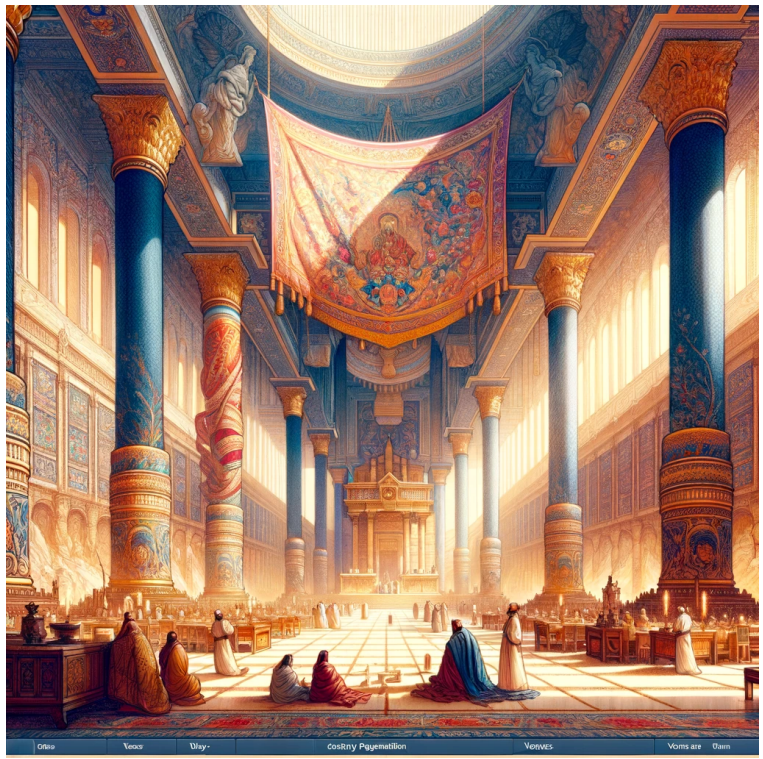


Figure: Views analogy

Models

Models in Rails are akin to the foundation stones of Solomon's Temple, underpinning the structure and determining its stability. They interact with the database, representing the temple's storerooms where records and treasures are kept.

Figure: Models analogy

Generators

Generators in Rails serve as the craftsmen and builders of Solomon's Temple, providing the tools and templates to quickly construct parts of the temple. They help developers by generating code snippets for different parts of a Rails application.

Bundles and Installs

Bundler and installation commands in Rails are like the supply lines and laborers bringing materials to the construction site of Solomon's Temple. They ensure all necessary tools and resources are available to build and maintain the temple.

Figure: Bundler analogy

Migration

Migration in Rails resembles the architects and scribes of Solomon's Temple, detailing how the temple's structure changes over time. They help modify the database schema without losing data, akin to planning renovations or expansions of the temple.

Figure: Migration analogy

Seeds

Seeds in Rails can be compared to the initial offerings and blessings in Solomon's Temple. They populate the database with initial data, setting the foundation for the temple's operations.

Figure: Seeds analogy

Scaffold

Scaffold in Rails is like constructing a temporary structure around Solomon's Temple to aid in its construction. It quickly generates the basic structure of a new section in the application, providing a framework on which to build further.

Figure: Scaffold analogy

Devise

Devise in Rails acts as the temple guards of Solomon's Temple, managing authentication and keeping unauthorized visitors from sacred areas. It provides a robust system for user authentication and management.

Figure: Devise analogy

Active Storage

Active Storage in Rails is akin to the storerooms of Solomon's Temple, where precious artifacts and scrolls are kept. It handles file uploads, allowing applications to store and retrieve various types of media efficiently.

Figure: Active Storage analogy

Conclusion

Just as Solomon's Temple was carefully structured to guide and facilitate its users, Ruby on Rails routes carefully guide requests to their proper destinations, ensuring the application functions as intended.