Programming Reference

LEAP Data Management Tool

Service Learning Practicum 2014

Document Version 1.0

Overview

The system was built using Rails on Ruby along with a MySQL database backend. The framework follows the Model-View-Controller (MVC) structure. The implementation logic is within the controllers. The relationships and entity properties are defined within the model and the page layout is within the view.

In addition to the programming reference, the Ruby code also contains comments that explain how the system operates in more detail. Comments before each method describe how the functionality was implemented and what role the method serves. Additional comments exist within the methods to help explain any advanced functionality.

Model Relationships

The system models encapsulate the MySQL data and represents relevant entities such as users, LEAP properties, as well as recordings.

Properties

Properties represent physical housing that LEAP gathers data on. The model contains key information such as the address and the owner's name.

Installed Measures & PropertyMeasure

Installed measures represent the energy-saving installations a house has gone through. A Property can have multiple installed measures. The PropertyMeasure model is the relational model between Property and Measures. The model simply contains a string describing the installation such as "Wall Insulation".

Recordings & RecordLookup

A Property can also have many Recordings. A Recording represents a month's worth of energy readings. The RecordLookup model ties Recordings to Properties. The RecordLookup model contains information on company name because Recordings can come from various companies like Washington Gas or Dominion. RecordLookup is tied to Recordings by referencing a Recording account number. When the system pulls data regarding a property, it refers to RecordLookup for account numbers tied to the property. The system then finds all associated Records that have the same account numbers.

Database Design

The MySQL database design is similar to the system's model designs. A set of tables in the database were designed to represent data regarding properties, users, recordings, etc. Another set of tables help define define the relationships between entities.

There are sets of SQL views that are defined for the customers that are ready to analyze (approximately 20-24 continuous utility data points for the customer) for each utility and views defined to request data for specific date ranges for each customer by utility company name. These views are created as migrations, migrated to MySQL, and accessed through the Report module. Inside each report module is the associated ActiveRecord object with each SQL view so it can be accessed by the controllers and views in the application.

External Components

The system utilizes a variety of gems and plugins for it's functionality.

Bootstrap-sass gem

This gem was used to generate the Bootstrap CSS and Javascript files which were used to style the website. The gem generated files in both the *app/assets/stylesheets* and *app/assets/javascripts* directories. It created a *framework_and_overrides.css.scss* file in the *stylesheets* directory. Also in the *stylesheets* directory, the *application.css.scss* file will be compiled into *application.css* and include all of the other CSS and SCSS files used for styling which are also in that directory. It also updated the *application.js* file in the *javascripts* directory to require bootstrap.

Devise gem

The Devise gem was used to create the MVC components which support user management. It ensures that only authenticated users can access the system. However, the Devise gem was unable to support some of the functionality required by LEAP. Therefore, to support only existing users creating and destroying other users, the Devise views had to be generated, the sign-up functionality taken away from the login screen, and different views and controllers created. The view that generates the login screen is in the *app/views/devise/sessions/new.html.erb* file. The files that support the functionality of registering a new user are the *app/views/newusers/index.html.erb, newusers_controller.rb,* and *config/routes.rb* files. The files that support the destroying of users are the *app/views/newusers/index.html.erb* and *app/controllers/users_controller.rb.* Those views and controllers are commented to detail the functionality.

Searching and Typeahead

The search bar is used to search for specific LEAP properties. Because multiple LEAP properties may have the same user name, the search bar requires both an owner name input as

well as address input. The search bar then queries for a specific property based on the owner name and address provided. To pass two inputs through the search bar, the system parses for owner name first and then the address separated by a hyphen. To simplify the process, the system utilizes typeahead/autocomplete. The typeahead functionality in the search bars was implemented using Twitter Typeahead, specifically their javascript files. The javascript was used in conjunction with Twitter bootstrap. A json array of all owner names and their properties is generated in /properties.json. Twitter Typeahead then utilizes this array to populate the suggestions list of the search bar.