GROUP-6 PS1: Documentation

Url: https://web06.cs.ait.ac.th/ (offline)
Gitlab repo: https://gitlab.com/ait-fsad-2022/web6/PS1

Features

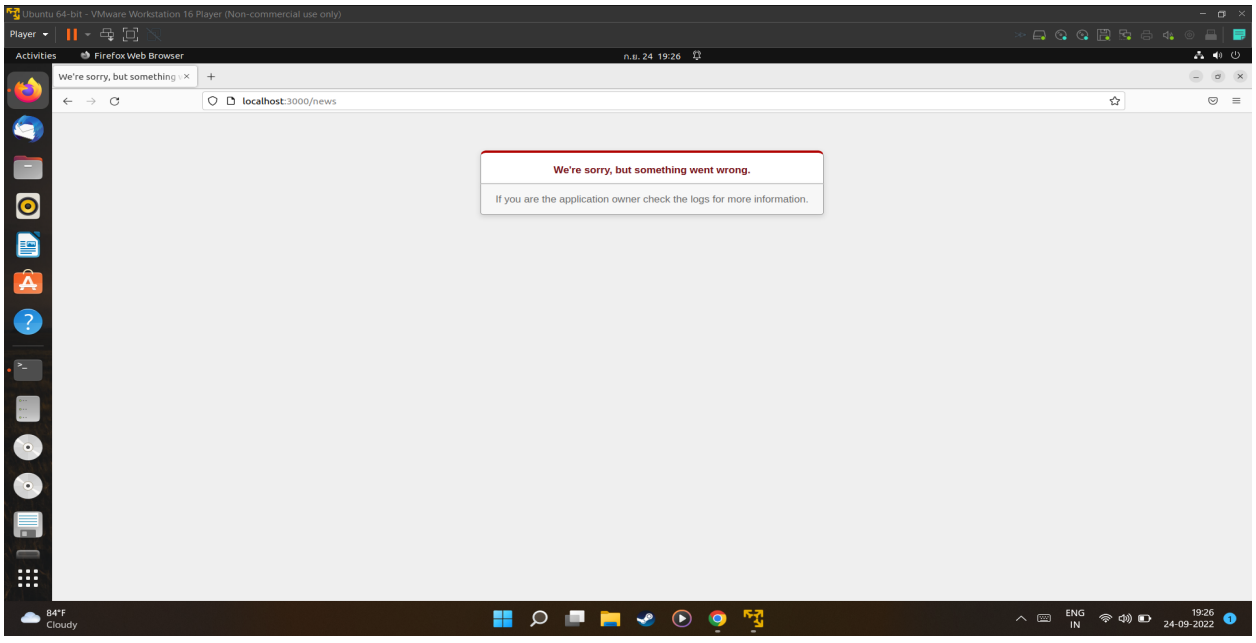
Adding XHTML Syntax Validation link to the bottom of pages

We created a _footer.html.erb file in the layouts folder. After adding relevant link to the XHTML Syntax Validation, we have inserted it in the application.html.erb file after the <%= yield %> syntax to display validation link in the bottom of the page.

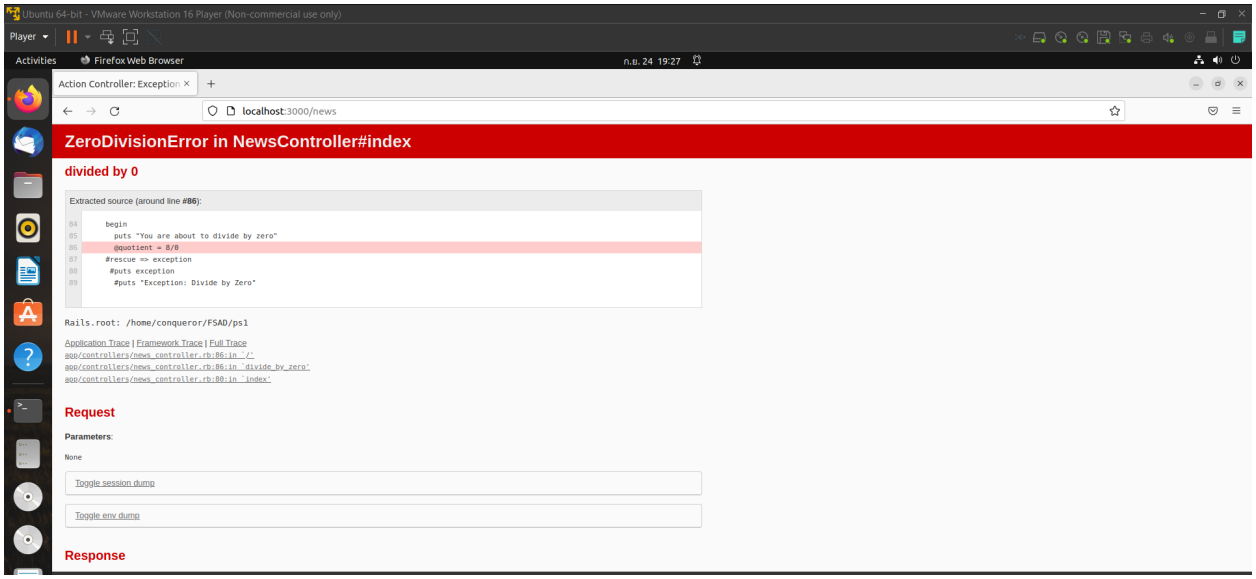
Reference link:
https://stackoverflow.com/questions/15308604/inserting-headers-and-footers-in-ruby-on-rails-web-application

Divide-by-zero Exception

We have added code that generates a ZeroDivisionError to the controller. After starting the server in the production mode, it displayed a generic error message but not the entrie stack trace.

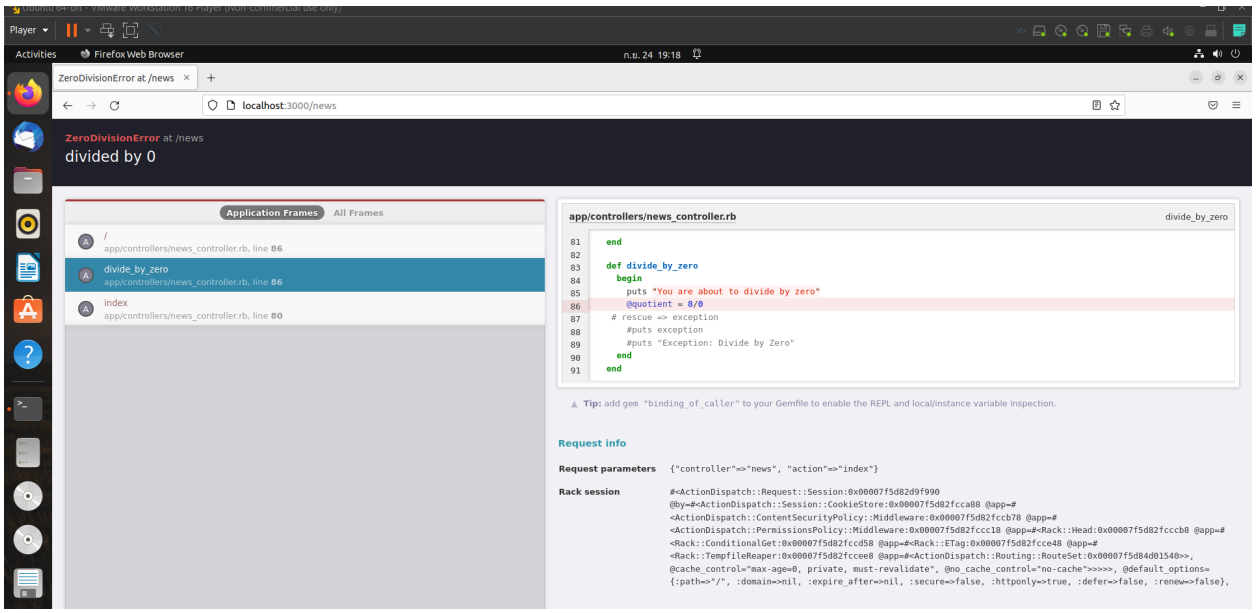


In order to output a stack trace to the browser, we have changed the **config.consider_all_requests_local = false** to **true**. This enables us to view the full error reports.



Since this is not considered as a best security practice to show error messages to the users, we have changed it back to false.

We have also tried to output the stack trace with another tool known as “**Better Errors**”. We have added gem ‘better_errors’ to the Gemfile and added it in the production group. This is the stack trace that is displayed by better errors.



We have added code that logs a message to the error log just before the ZeroDivisionError.

Reference link:
https://stackoverflow.com/questions/36670968/rails-4-exception-stack-traces-dont-show-in-development

http://railscasts.com/episodes/402-better-errors-railspanel?view=asciicast

Fetching news headlines

We created a **news controller** that fetches html data from https://www.nytimes.com/world and https://www.bangkokpost.com/world, parses it, extract title, excerpt and link and displays into the news views.
We used the gem httparty to facilitate this process. It sends get requests to the news pages and returns a response object.
To parse the response body used another gem called Nokogiri which makes it easier to select html elements based on tags, class names, attributes and so on. In this case for example we saw that each featured news had a parent div with class name *hilight_sec--list* and used that. Finally we used css selectors to select the h2, hrefs attributes from <a> contained within that parent div.

Root of our app is set on site controller site#index, and we added a link there pointing to /news view

Reference link: https://oxylabs.io/blog/web-scraping-with-ruby