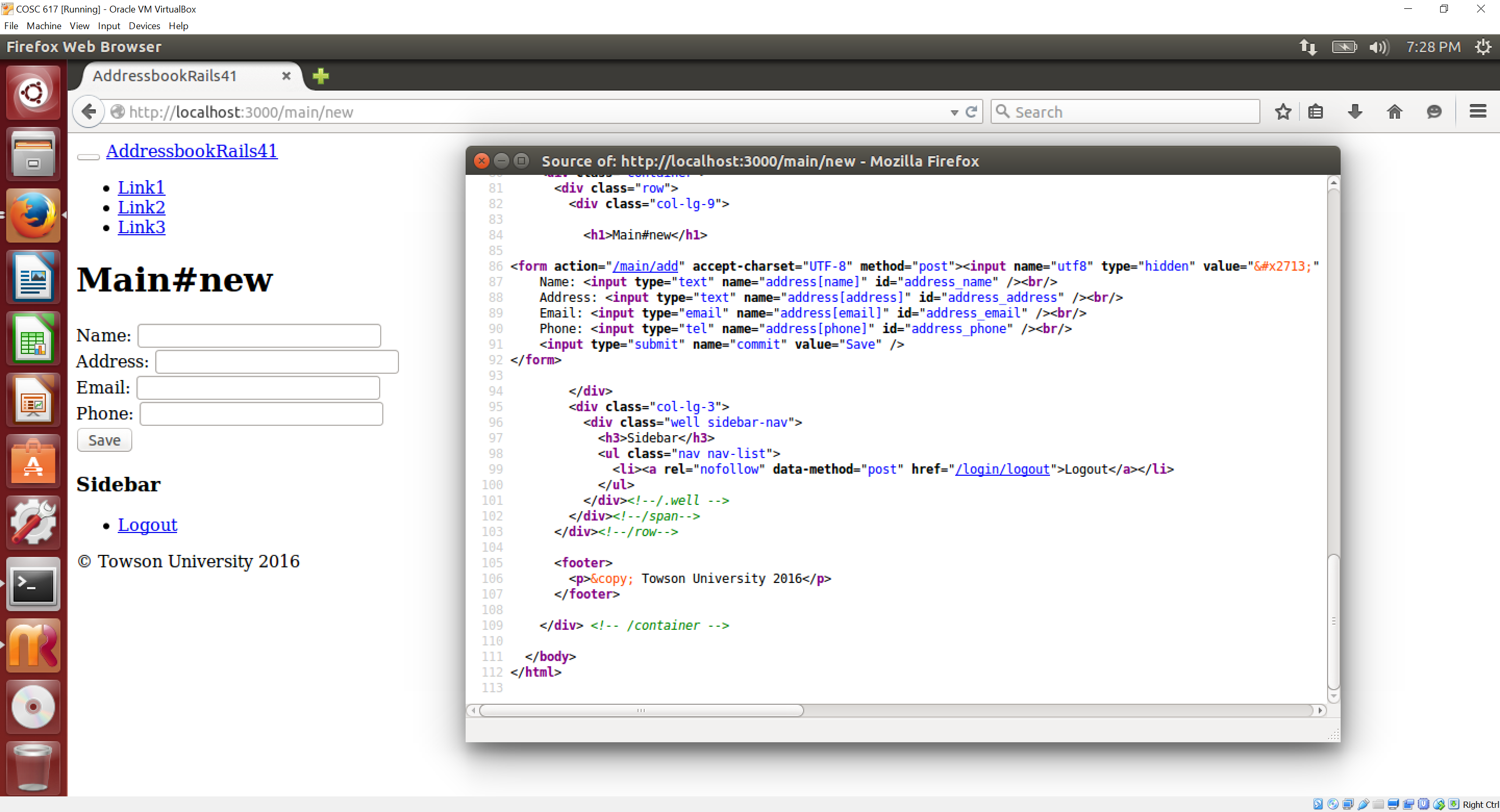
**Advanced Web Development – Basic Forms and Bootstrap**

*Kevin Kuo and Mary Snyder*

Due date on blackboard

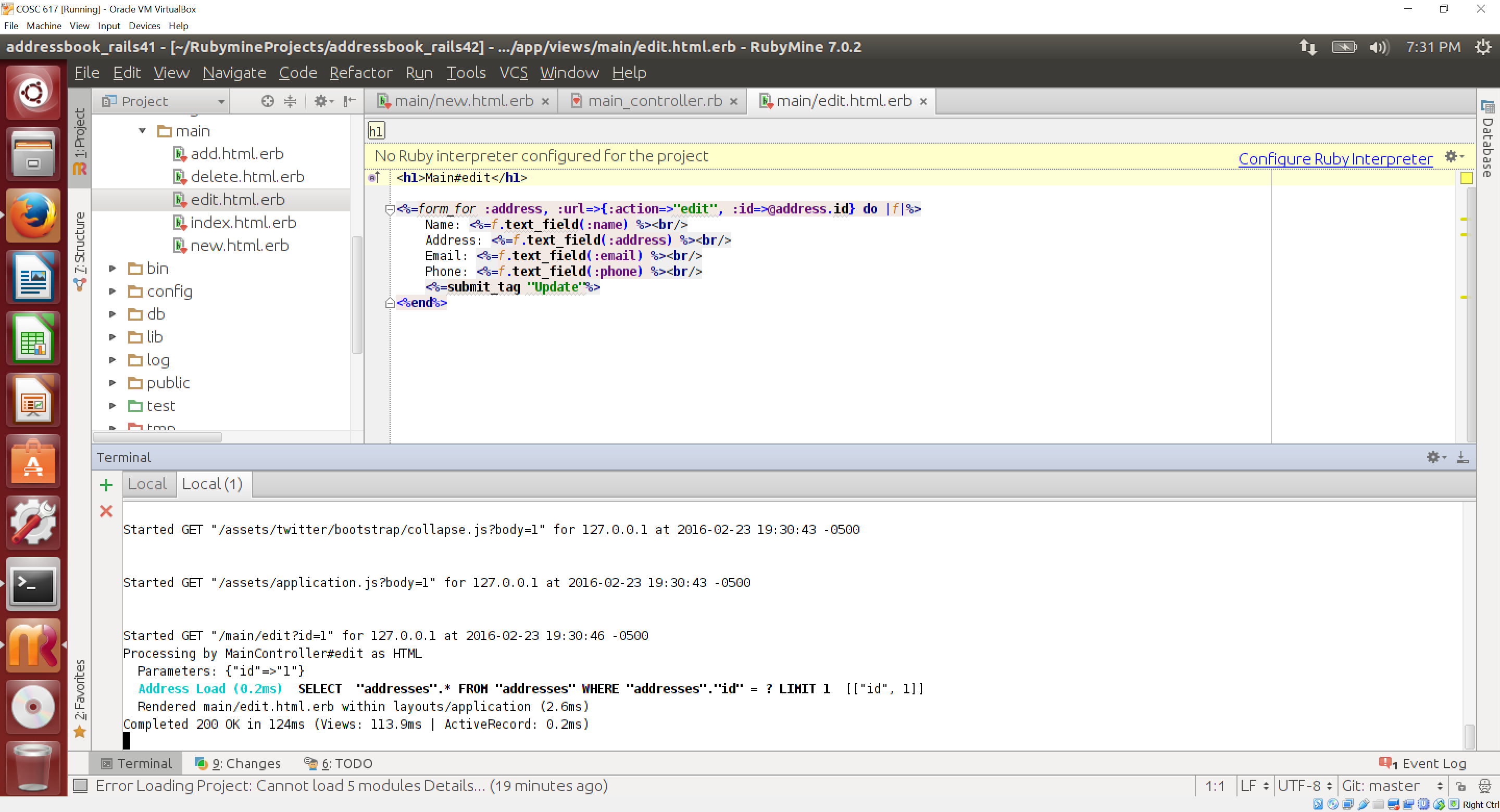
1. Download the addressbook example form the blackboard under today’s lecture (the non-bootstrap version) and get it running. (10 points)
   1. Run the application and view the main#new view in the browser. Open the source of the HTML (possible in all browsers, you can also use a developer tools – in most browsers). See the source of the form generated. You will notice that there is a <input type=email>. What does that accomplish? *Hint: This is HTML5 functionality.* How would this benefit a person viewing this page on a mobile device?

**The <input type=email> is used to check certain patterns to ensure a valid e-mail address was entered. This might be beneficial for a person viewing on a mobile device because some phones display a custom keyboard for the <input type=“email”> with dedicated ‘@’ and ‘.’ characters to help the user fill out the field more efficiently.**

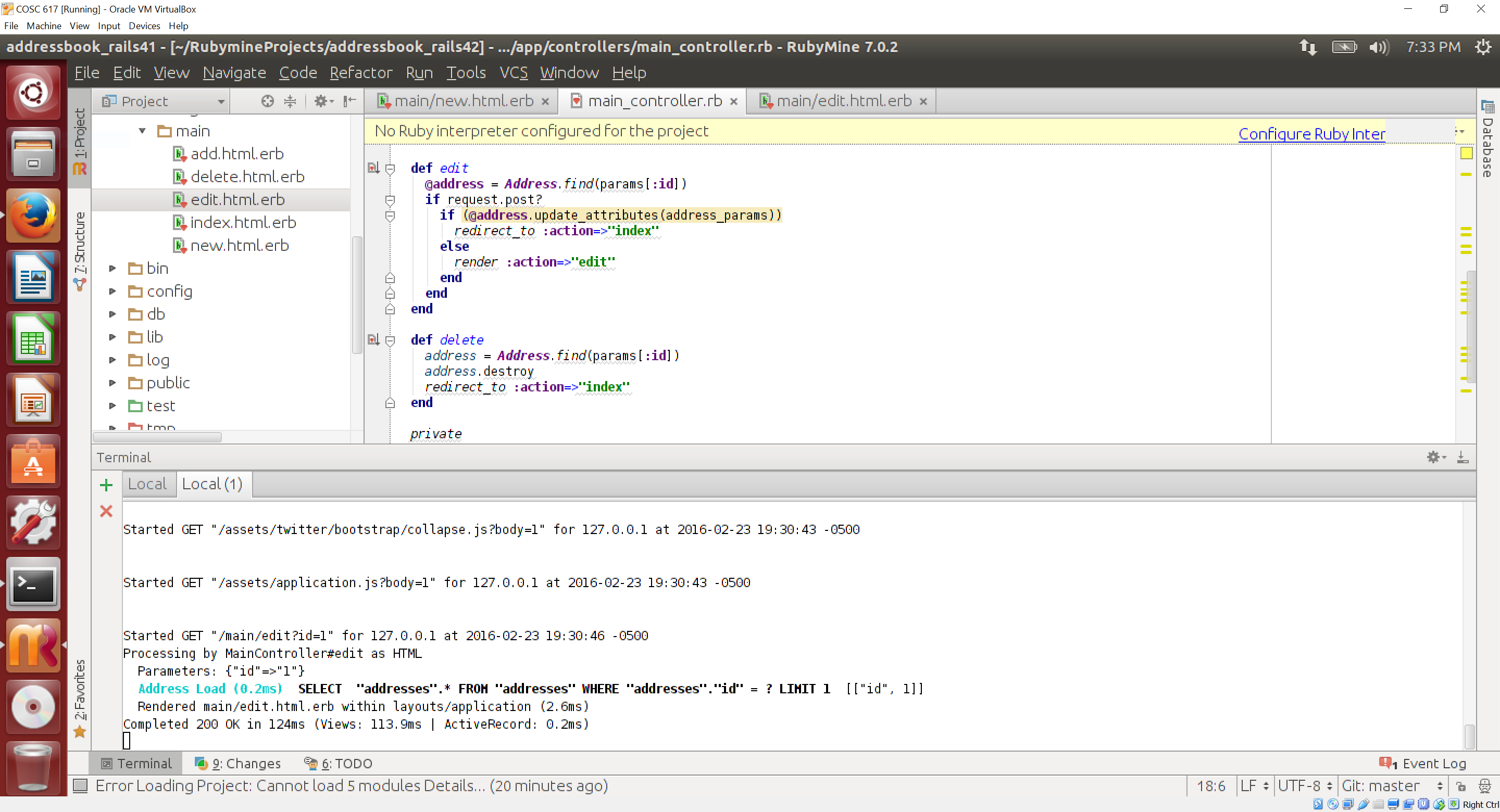


* 1. Press the “edit” link next to a record in the address listing. Find the parameters that are passed in the post request (you may have to look at the log file). What is the data structure that is passed to the controller?

**Parameters passed into the post request are id=1. The data structure passed to the controller is the record from the database that matched the passed in id.**

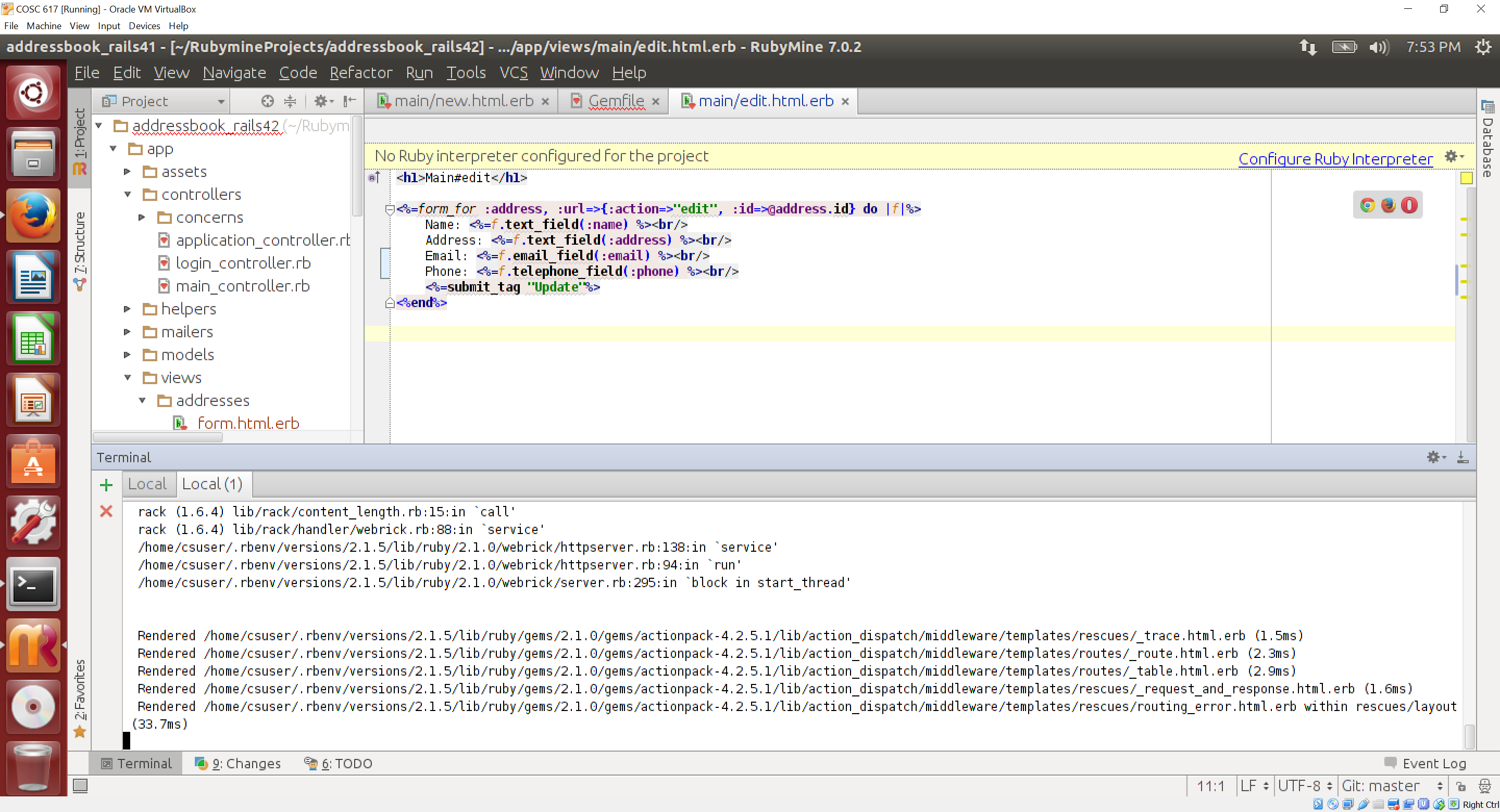


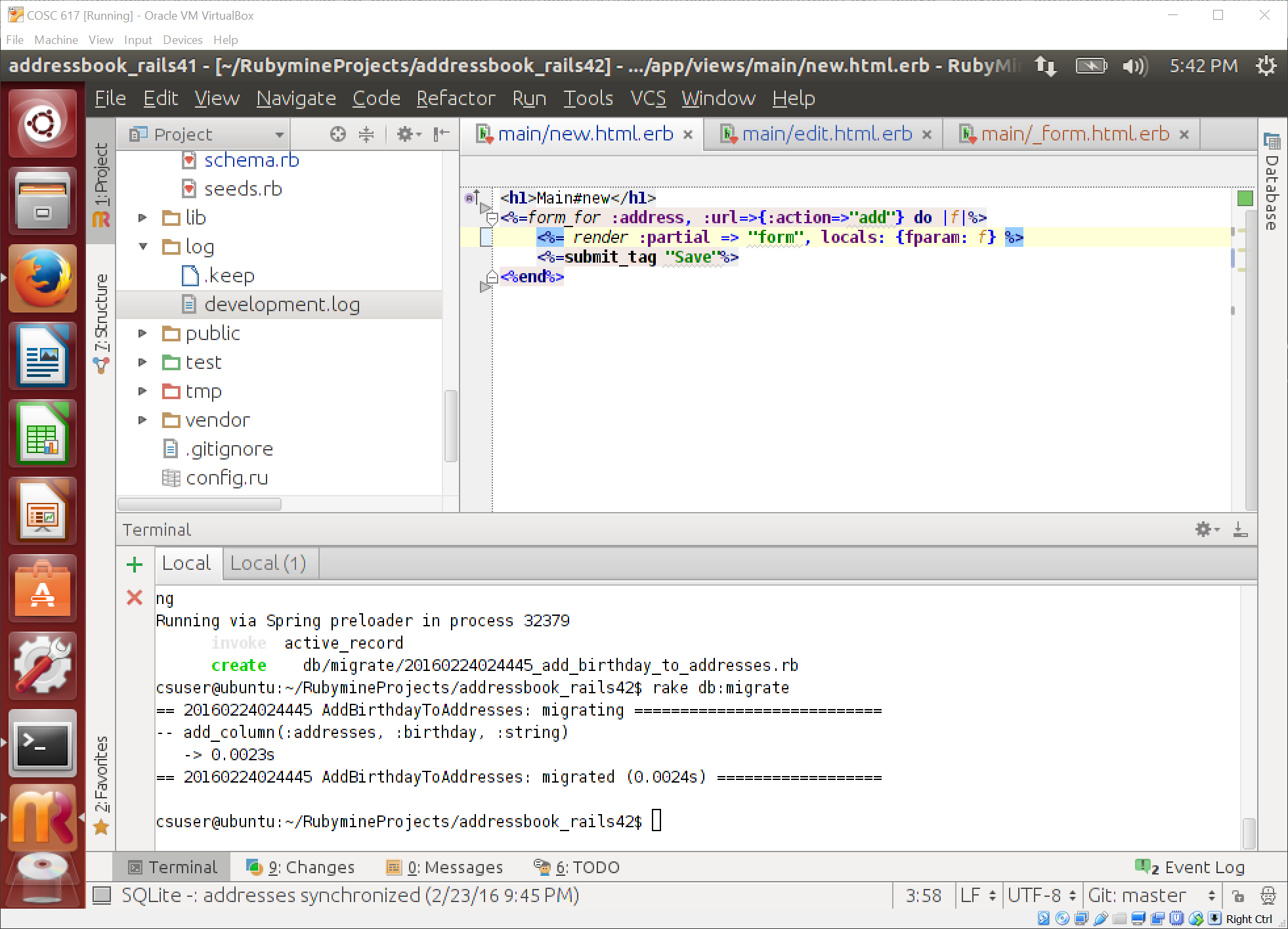
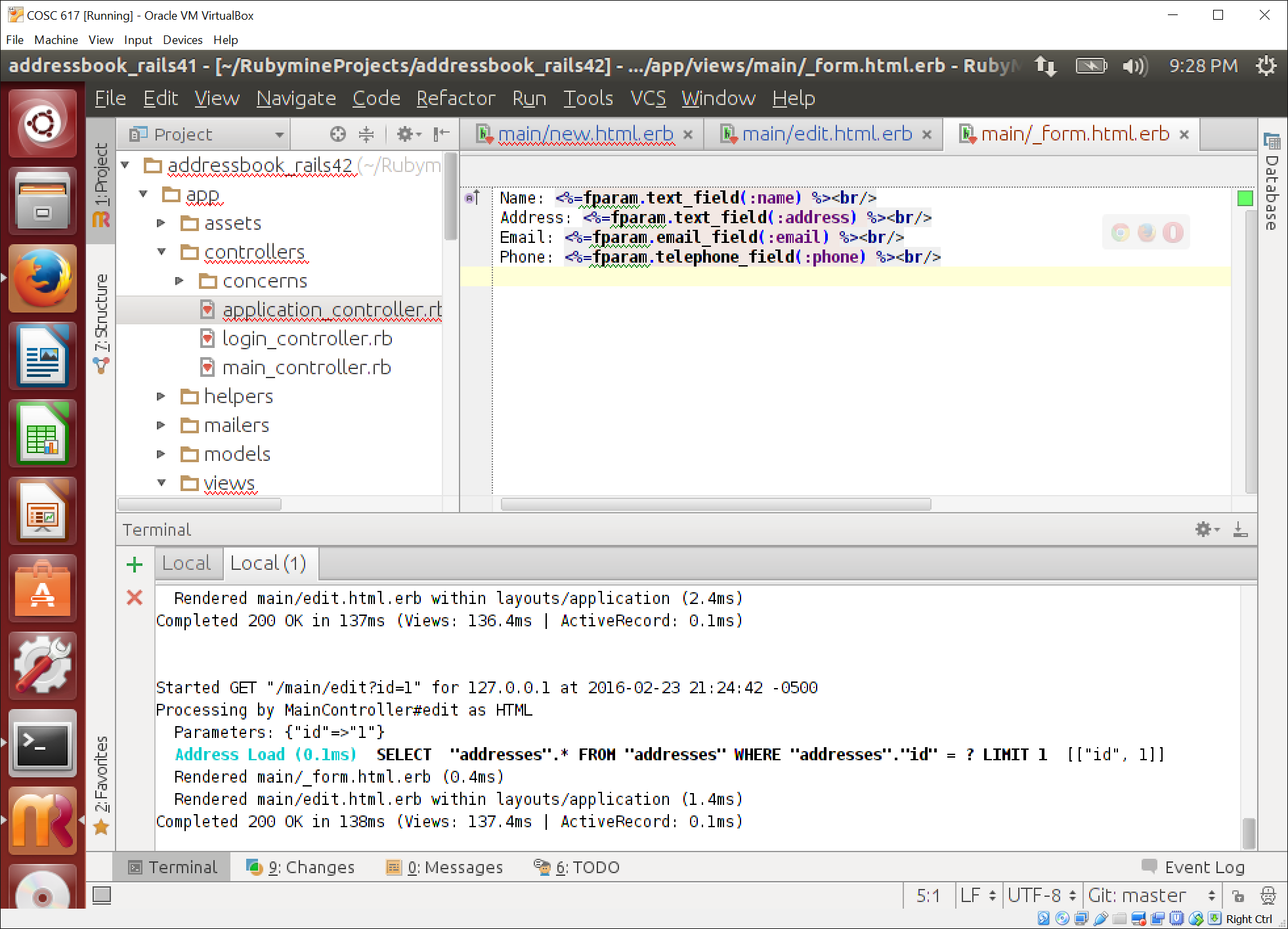
* 1. Look at the main#edit action in the controller. Why do we check if the request is a ‘post’ vs a ‘get’?

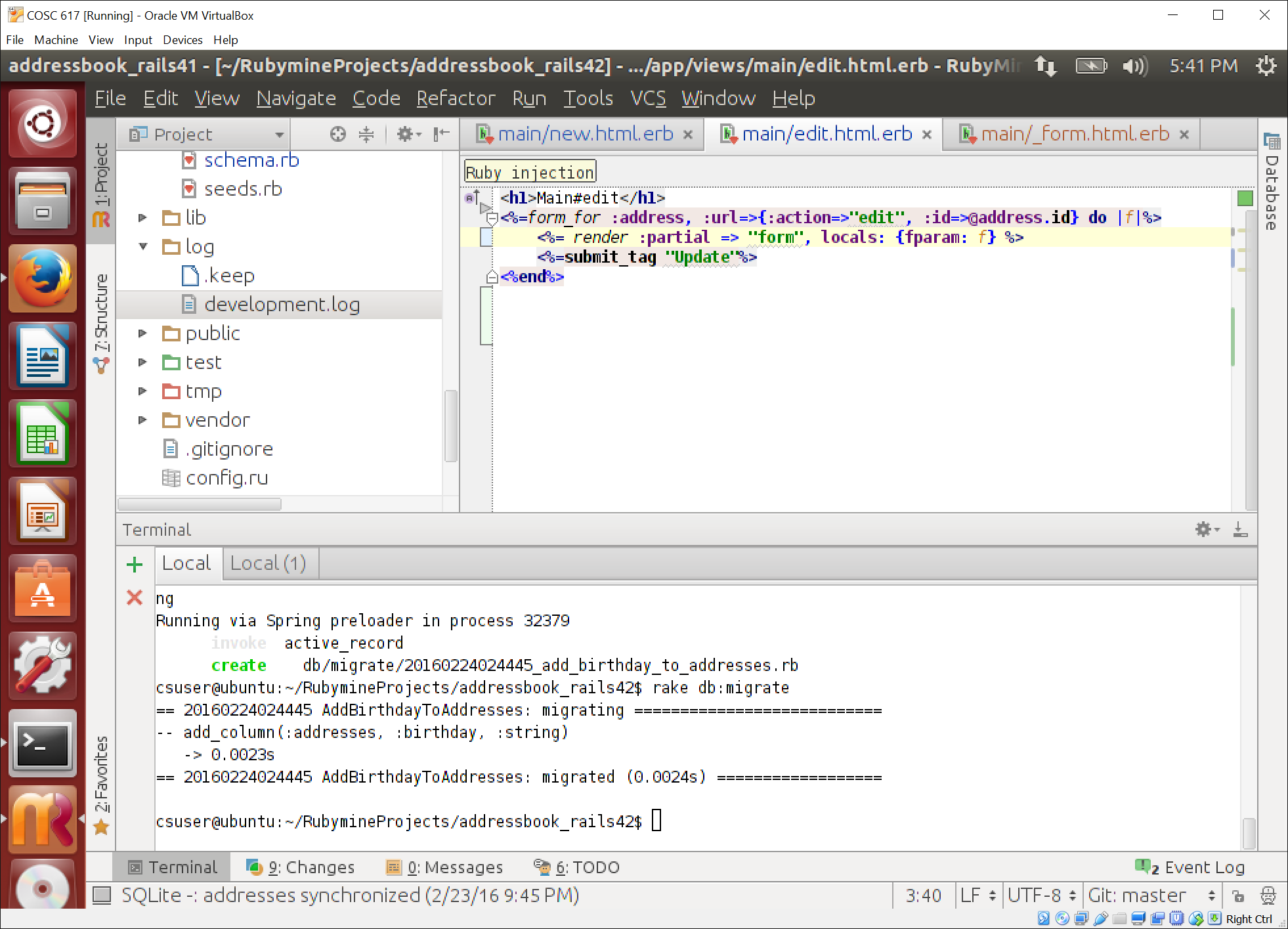


**We check if the request is a ‘post’ vs a ‘get’ command to know if the user is done editing or beginning the edit. A ‘get’ command is used when the edit is initiated to obtain the record to be edited from the database and populate the form text fields for the user to change. The ‘post’ command is used when the user has completed their edit and wants the updates/changes from the form committed to the database.**

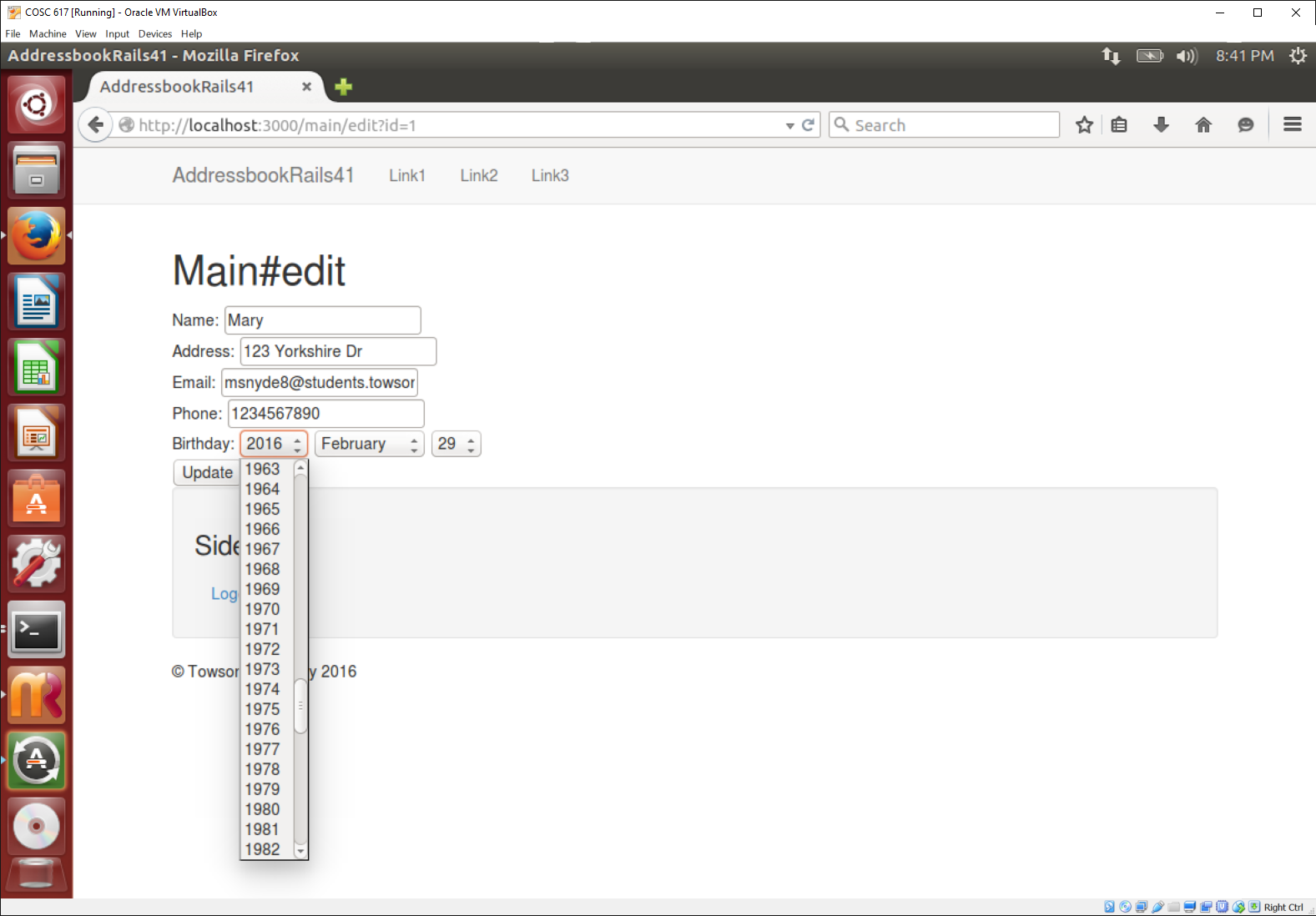
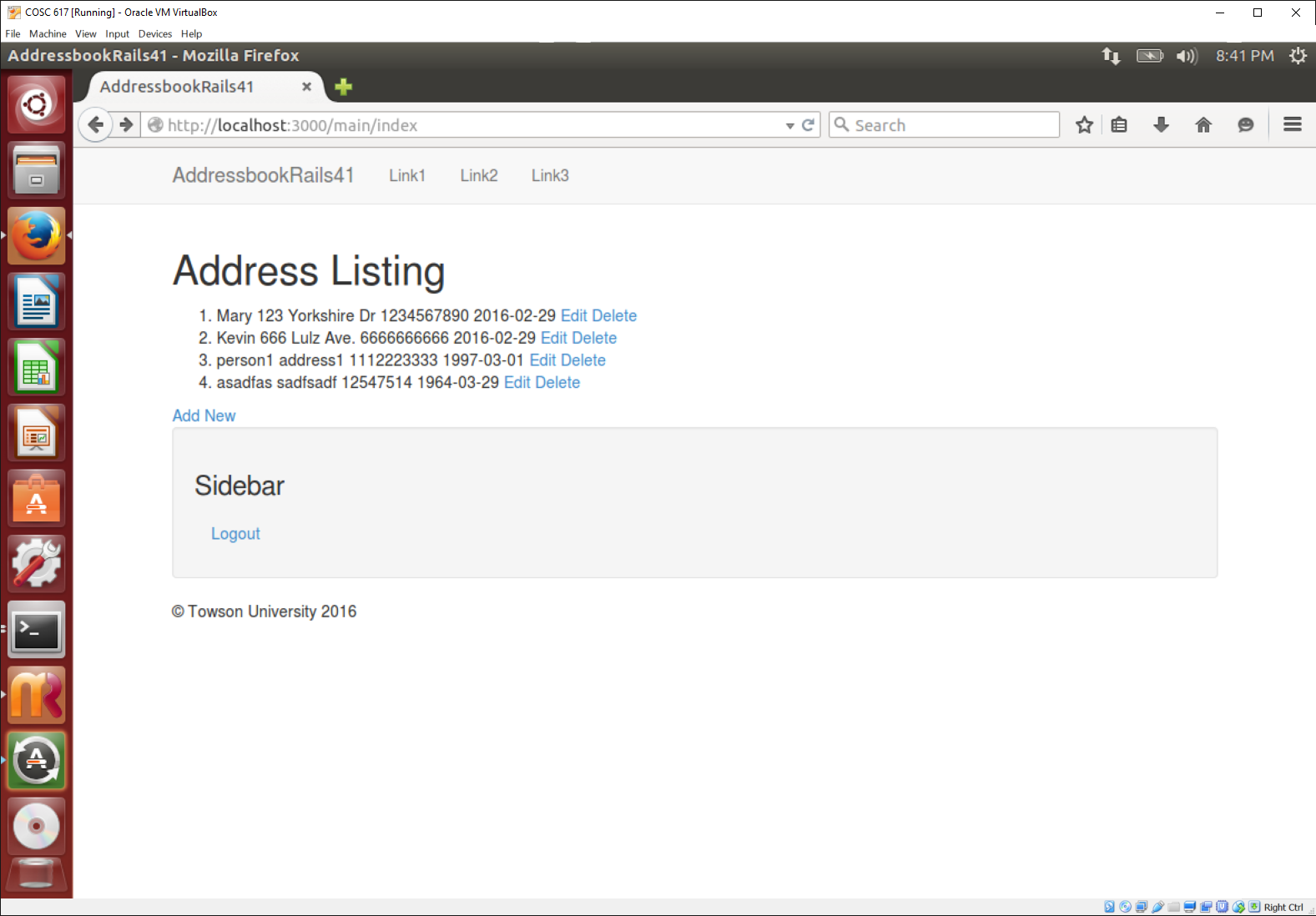
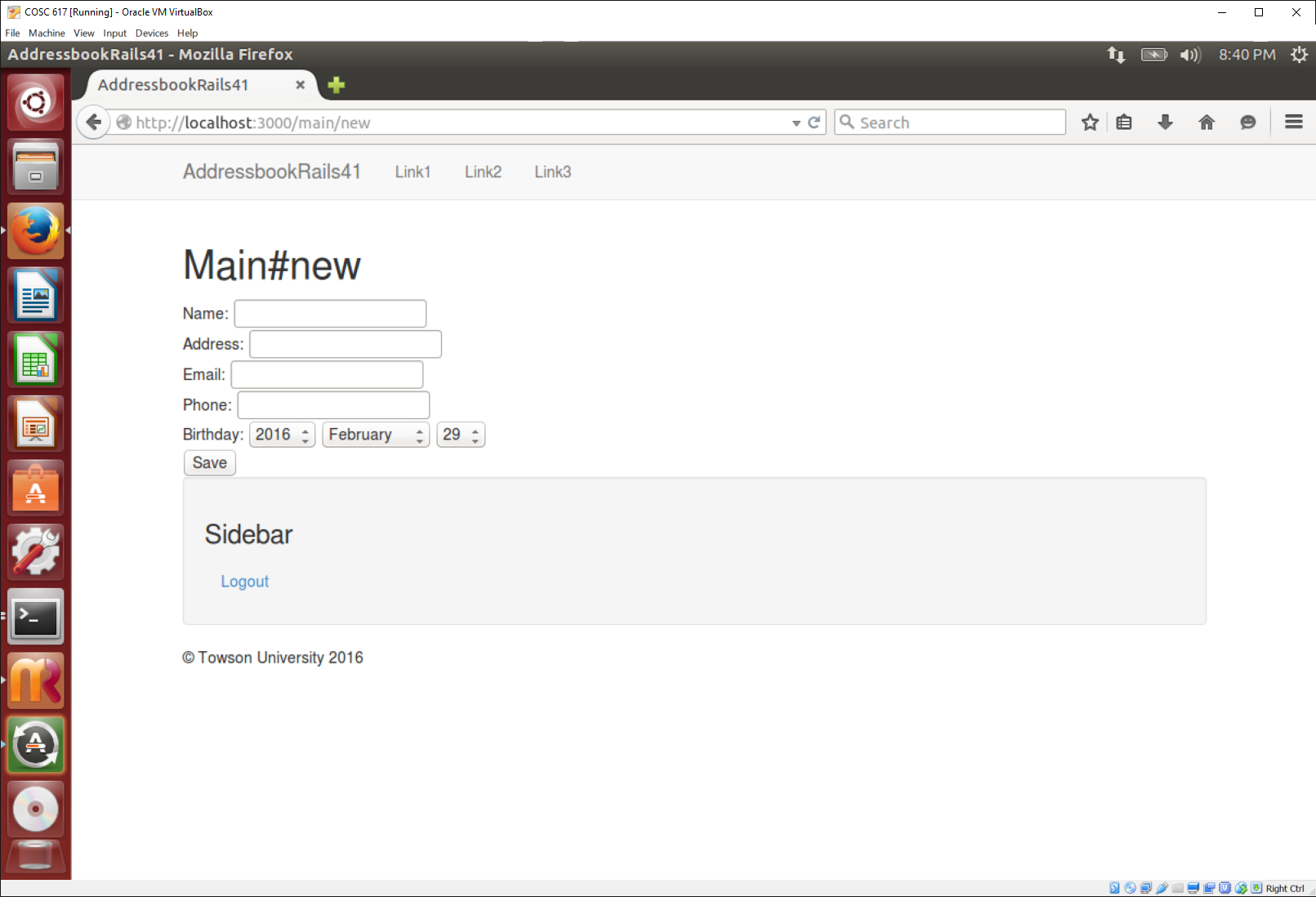
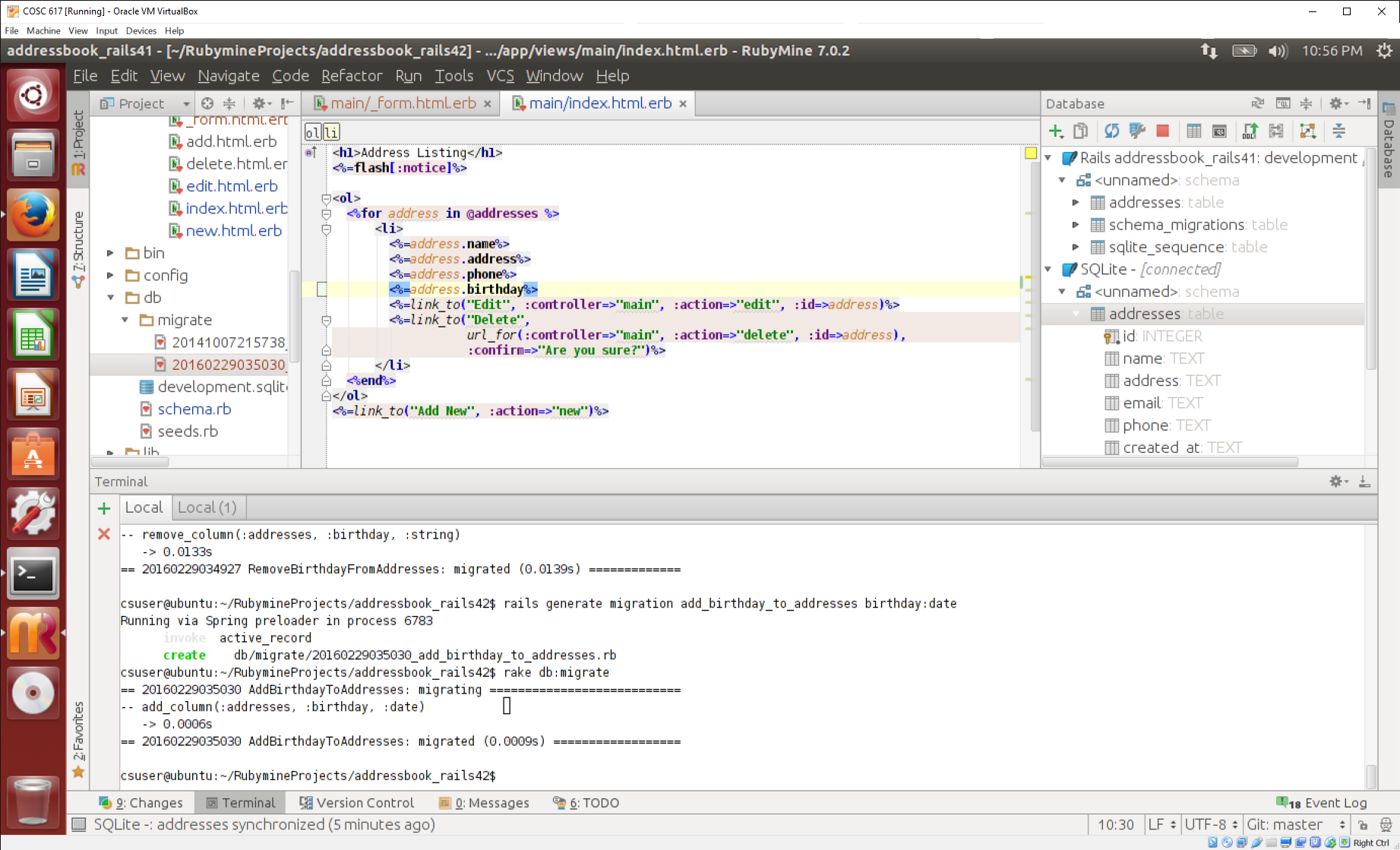
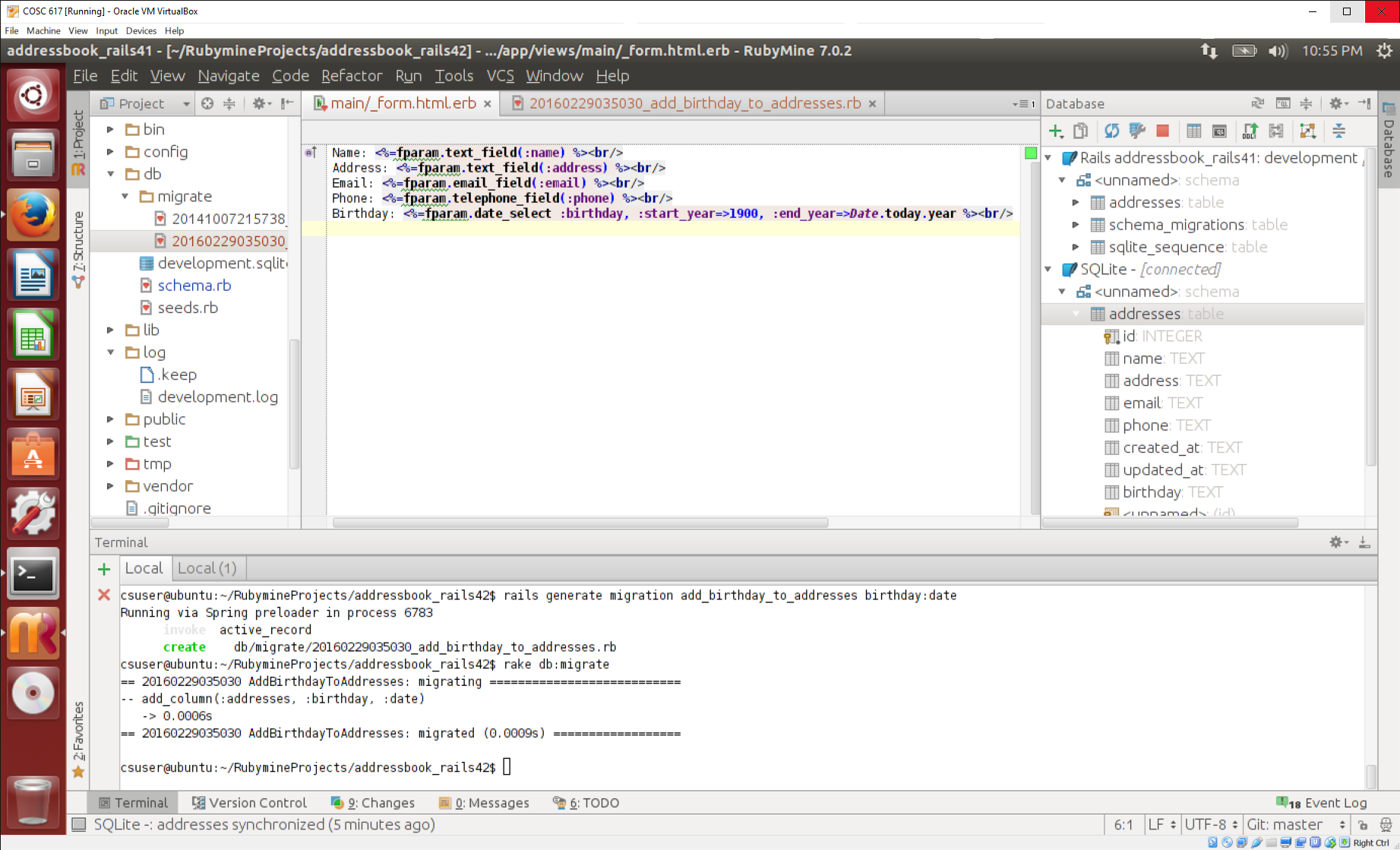
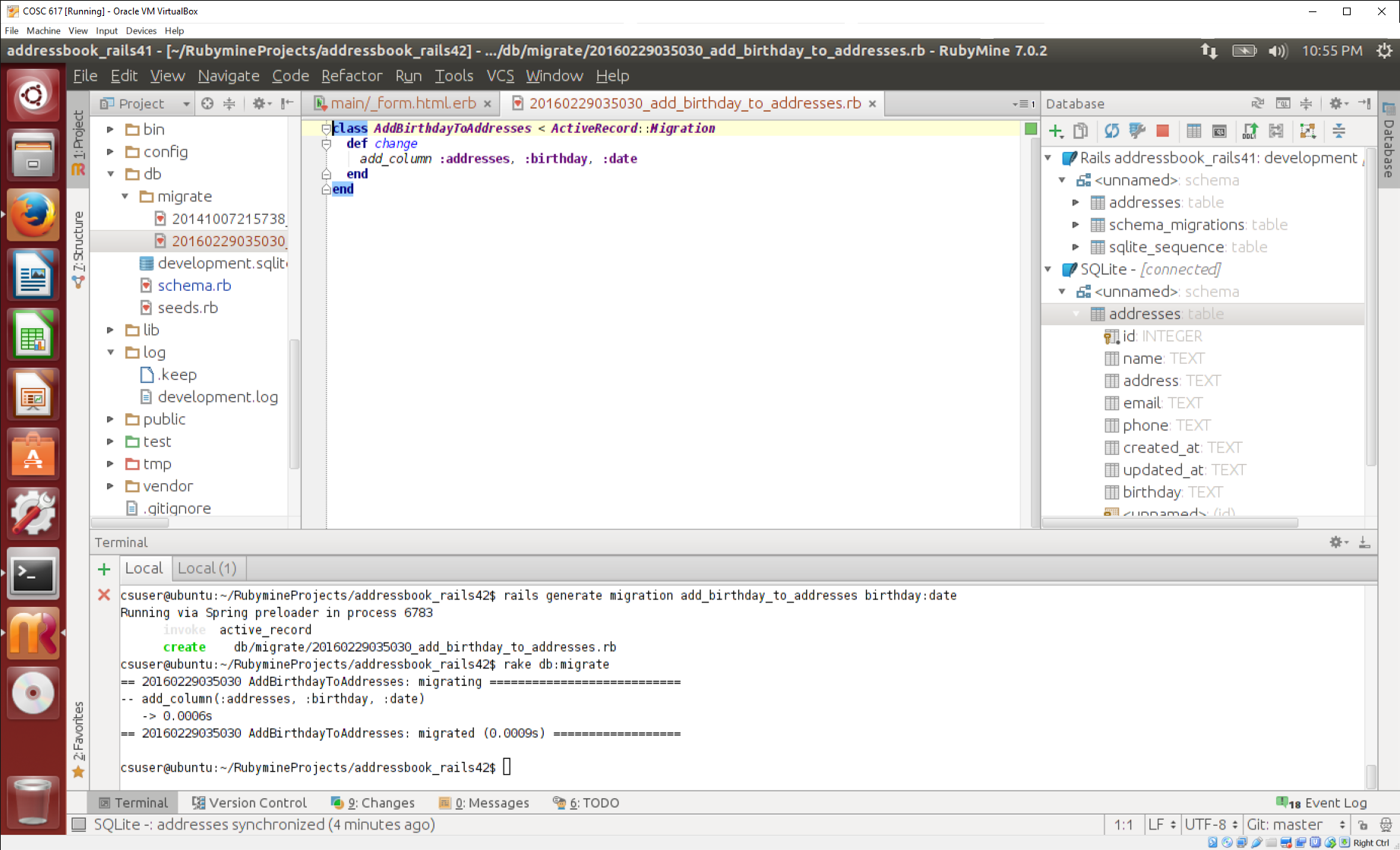
* 1. Notice in the main#new view, I have used the email\_field() and telephone\_field() helpers. These are not present in the main#edit view. Please add them there.



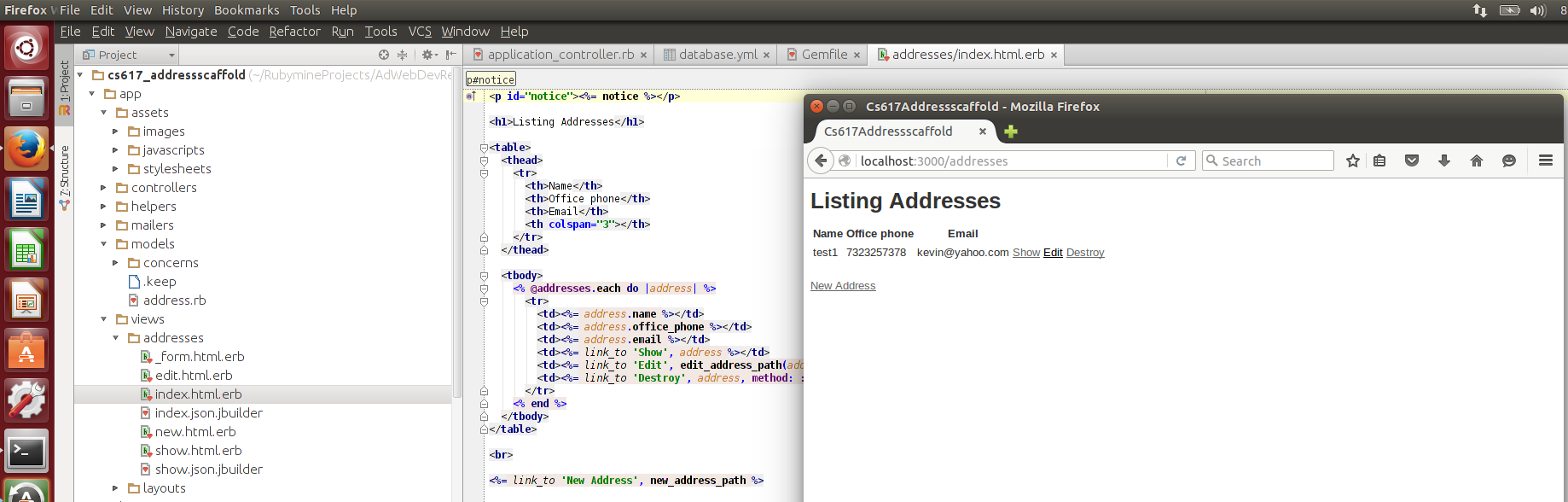
* 1. Notice the duplication of effort here, you have the same code repeated twice. DRY! Rails allows the use of partial views (read <http://guides.rubyonrails.org/layouts_and_rendering.html>, this is also in the AJAX chapter in the book). Replace both main#new and main#edit to use a partial called \_form.html.erb. You will have to create the partial and modify both files. Show a screenshot with partial here. 



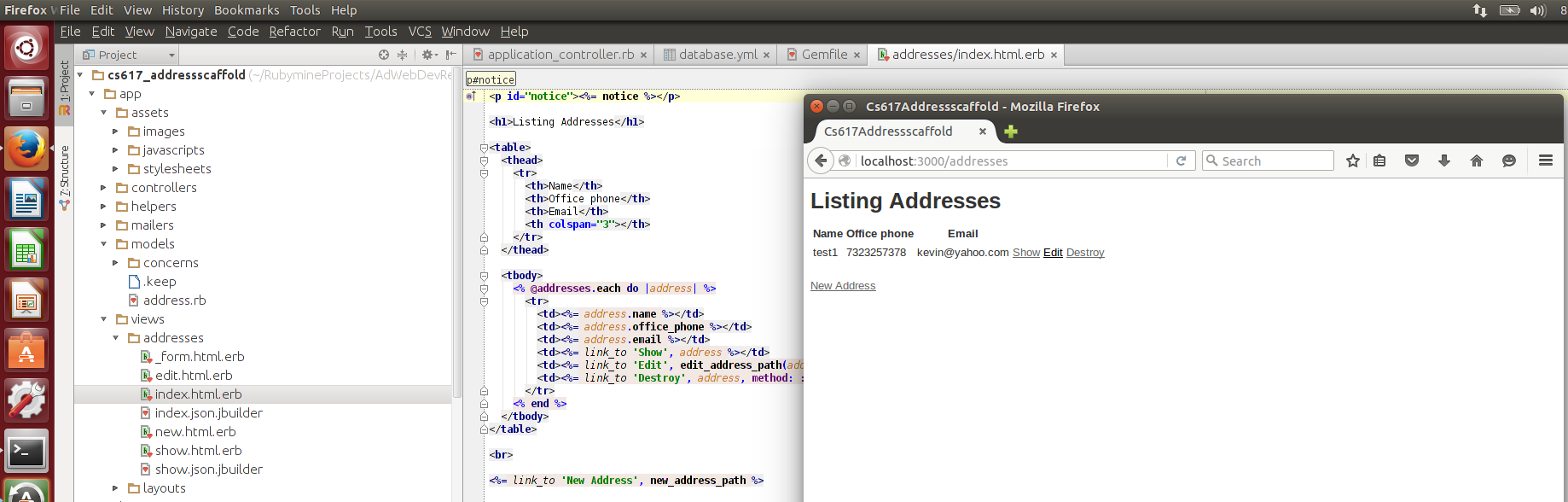
* 1. I want to record the birthday of every contact. Change the code so it accepts and stores birthdays. Make sure you use the date\_helper (see here <http://guides.rubyonrails.org/form_helpers.html> ).

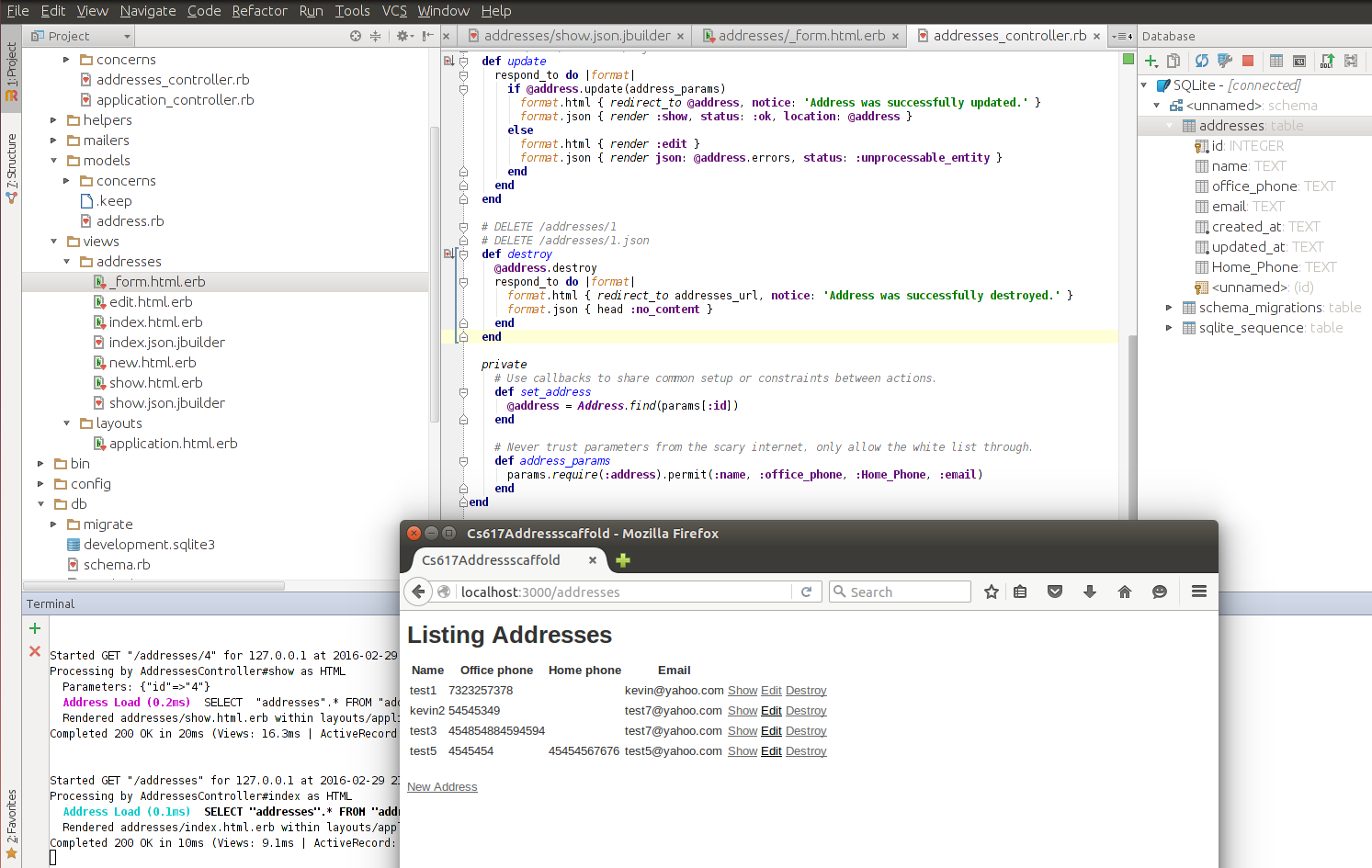


1. Scaffold the addressbook (10 points)
   1. Make a new application for the web address book that uses the ‘scaffold’ functionality. The address model should contain name, office\_phone, and email.



* 1. Once you have the application running, add some data to the database using the web form.



* 1. Now, you have to add the ‘home\_number’ field to the database. You will have to modify several files to make this work since you used a scaffold. Modify those files and show a screenshot showing a listing of addresses with the new field. 

3. Create an application, bootstrapped, committed to version control (20 points)

1. 10 points. Create any scaffolded application (lets say a to-do list application that organizes your todo list and allows you all the CRUD operations on the todo list) and add bootstrap to it. I used the rails-twitter-bootstrap gem (<https://github.com/seyhunak/twitter-bootstrap-rails>). A demo of the addressbook application is available in the lecture resources. Design this application so it is workable on a mobile device (pick appropriate styles in bootstrap). There should be seed data in the DB.
2. 10 points. Change the RDBMS of the application to postgres. This might mean installing postgres on your machines/VMs. You will need to migrate and seed the databases again and almost certainly change the database.yml file. If you don’t succeed in changing the current application then create a new application from scratch to use postgres.
   1. Explain the ‘development,’ ‘production,’ and ‘test’ portions of the database.yml file.

**The database.yml file allows the user to specify all the information needed to access the database. The ‘development’ portion of the file is used to specify the environment for the local (development) computer to use when interacting with the database manually. The ‘production’ portion of the file is used to specify the environment when the application is deployed for the rest of the world to use. The ‘test’ portion of the file is used to specify the environment for running automated tests.**

**Submission:** Paste screenshots of the desktop view and mobile view. You can use the Chrome plugin to test the mobile/responsive views - Check out "Mobile/Responsive Web Design Tester": <https://chrome.google.com/webstore/detail/mobileresponsive-web-desi/elmekokodcohlommfikpmojheggnbelo?utm_source=gmail>

Provide the URL to your repository.