

## Assignment 3

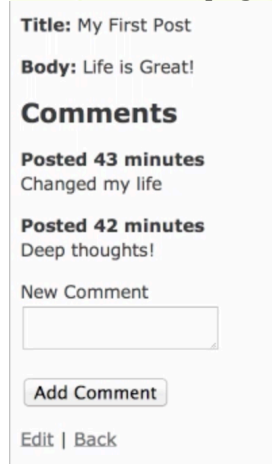
Due Date: December 11, 2013

For this assignment students will be adding additional features to their blog application to improve the “user experience” of the application.

Note: There was some miscommunication on the previous assignment as to what the requirements were. Due to this, students may be at slightly different stages on this assignment. To complete the assignment, follow the generalized instructions below and watch the blog iteration videos posted on learn.

### 1. Add new comment form underneath post page

Up till now, your new comment form has been on a page of its own. Now move that form into the post show action. It should look similar to the image below. Additionally, make sure that the “Add Comment” button is redirecting to the current page.



The screenshot shows a blog post titled "My First Post" with the body text "Life is Great!". Below the post content is a "Comments" section. It lists two comments: one posted 43 minutes ago with the text "Changed my life", and another posted 42 minutes ago with the text "Deep thoughts!". Below the list of comments is a "New Comment" section containing a text input field and an "Add Comment" button. At the bottom of the comments section are links for "Edit" and "Back".

### 2. Make the form remote! And render JS

Now that you have a form on your post show action, let's try to make it more user friendly and visually pleasing. To do this, you will need to modify the new comment form to perform the POST request remotely.

To see how to make a form function remote, follow this link (it is under the 3<sup>rd</sup> section “Built-in Helpers”):

[http://edgeguides.rubyonrails.org/working\\_with\\_javascript\\_in\\_rails.html](http://edgeguides.rubyonrails.org/working_with_javascript_in_rails.html)

Now if you test your application no redirect will take place when you submit a new comment. However, this also means you will need to do a manual refresh of the page to see the comment that has been added remotely.

To avoid the manual refresh we will need to add a javascript response to the comment create action. Inside the comment create action, there should be the following if statement:

```
if @comment.save
```

Inside this condition, html and json get rendered, but not javascript. Simply add "format.js" inside the "if" statement. This causes the show.js.erb file to be rendered; you will need to create it within the comment view folder.

This javascript file will be used to inject the new comment into the current page with your choice of animation (fade, slide-in, etc.).

Explore the following link on how to use jQuery to append html to an html element:

<http://api.jquery.com/append/>

### 3. Styling

Now that you have a barebones application setup, try your hand at styling this app. Take a look at some popular blog sites and see if you can implement some styles that you like into your own blog. These may include, but are not limited to, adding a site header, indenting the comments section underneath a post, sectionize the elements in the page to make it more readable, etc.

### 4. Submission

I would like all of you to demon straight the blog app to me in person if you are able to. If unable, you may just submit it through BitBucket, but make sure you tell me by email that you won't be able to personally show me ([kwarne@unm.edu](mailto:kwarne@unm.edu)). I will be on campus December 10<sup>th</sup> and 11<sup>th</sup>, from 1:00 PM to 4:00 PM in ECE ROOM 218A. If you need another time, please email me.

### 5. Extra Credit

For extra credit you may add users and authentication to your application. For this you must create a User table to your database structure. Then add the Devise gem for Users to sign in to their account. Finally a "has many - belongs to many" association will need to be added for Users to Posts and Users to Comments. Finally when a User is signed in, they will be able to only edit the comments and posts that they made.