EDIT QUESTION

In the previous lesson we've created a `CreateQuestionPage` component and mapped it in our `routs.js` file so that we can access it in our single page application. We've also created `QuestionForm` component as a child component that hold the actual form.

Now in this lesson we're going to create another page that allows us to edit existing question in our single page application. We're going to making use the same component that is `QuestionForm` as child component. So I think this task is more easier.

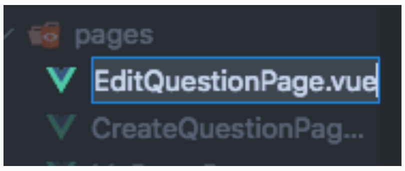
Alright, let's go ahead and open up our terminal then create a new branch.

```
git checkout -b lesson-60
```

After that let's run Laravel mix by running `npm run watch`.

CREATING EDIT QUESTION PAGE COMPONENT

Alright let's firstly create a new component that will represent an entire page. Navigate to `js/pages` directory. Then Inside this directory let's new file called `EditQuestionPage.vue`.



Since the logic for this component is almost the same with `CreateQuestionPage.vue` so let's also open that file. Then copy everything inside that file and paste in the `EditQuestionPage.vue`.

1, Modifying the template

Let's change the page title to `Edit question` in the `h2`. In the `question-form` component calling let's change the method call from `create` to `update`.

Also let's add another attribute to that component to indicate that the form is gonna be used to edit data. Let's call it `is-edit` and the value is going to be `true`.

```
<template>
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-12">
                <div class="card">
                    <div class="card-header">
                        <div class="d-flex align-items-center">
                            <h2>Edit Question</h2>
                            <div class="ml-auto">...</div>
                        </div>
                    </div>

                    <div class="card-body">
                        <question-form
                            @submitted="update"
                            :is-edit="true"></question-form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</template>
```

2. Modifying the script

Next, let's go to script section and define the `update` method. We can simply replace the `create` method definition in `methods` property to `update`.

In that method what we need to do is to change the request method from `post` to `put`. And change the request path from `'/questions'` to `'/questions/' + this.$route.params.id`. The `this.$route.params.id` will capture the `id` parameter from the URL.

```
<script>
import QuestionForm from '../components/QuestionForm.vue'
import EventBus from '../event-bus'
export default {
    components: { QuestionForm },

    methods: {
        update (data) {
            axios.put('/questions/' + this.$route.params.id, data)
                .then(({ data }) => {
                    this.$router.push({ name: 'questions' })
                    this.$toast.success(data.message, "Success")
                })
                .catch(({ response }) => {
                    EventBus.$emit('error', response.data.errors)
                })
```

```
          }
      }
  }
</script>
```

Now the next thing that we need to do is to make a little tweak in the `QuestionForm` component to populate the old data in the form input when editing the question.

MODIFYING THE QUESTIONFORM COMPONENT

1. Defining the isEdit property

So let's open `QuestionForm.vue` file then go to script section. Remember we've specified `isEdit` when calling this component in the `EditQuestionPage` component.

So let's add `props` property. Then define the `isEdit` property inside. This property is going to be `Boolean` and we'll make it `false` by default. By doing this way we don't need to touch the `CreateQuestionPage` component to explicitly specified this property.

```
export default {
  props: {
    isEdit: {
      type: Boolean,
      default: false
    }
  },
  // ...
}
```

Now we can making use the `isEdit` value to determine two things:

- Change the button's text
- Populate the form inputs

2. Change the button's text

In the `computed` property let's make a small change in the `buttonText`. Let's make the button's text to "Update Question" if the `isEdit` is `true`. Or "Ask Question" if the `isEdit` is `false`.

```
computed: {
  buttonText () {
    return this.isEdit ? 'Update Question' : 'Ask Question'
  }
}
```

3. Populating the form inputs

Now in order to populate the form inputs if the `isEdit` is `true` we need to hit the *Edit Question* endpoint. And we'll call that endpoint in the `mounted` hook.

```
mounted () {
  EventBus.$on('error', errors => this.errors = errors)

  if (this.isEdit) {
    this.fetchQuestion();
  }
},
```

We can then go to `method` property and define the `fetchQuestion` method. In this method we hit the *Edit question* endpoint using `get` request. In `then` method we can assign the `title` and `body` from api response to `title` and `body` property respectively.

```
methods: {
  // ...
  fetchQuestion () {
    axios.get(`/questions/${this.$route.params.id}`)
      .then(({ data }) => {
        this.title = data.title
        this.body = data.body
      })
      .catch(error => {
        console.log(error.response);
      })
  }
}
```

CREATING THE ROUTE RECORD

Alright, the next thing that we need to do is to add a new route record in our `router/routes.js`. At the top import the `EditQuestionPage` component. Then we can simply duplicate the existing route record.

The `path` is going to be `'/questions/:id/edit'`. So you can think of the `:id` as router param in laravel like this `{id}`. The `component` is gonna be `EditQuestionPage`. And last we can call it as `questions.edit`

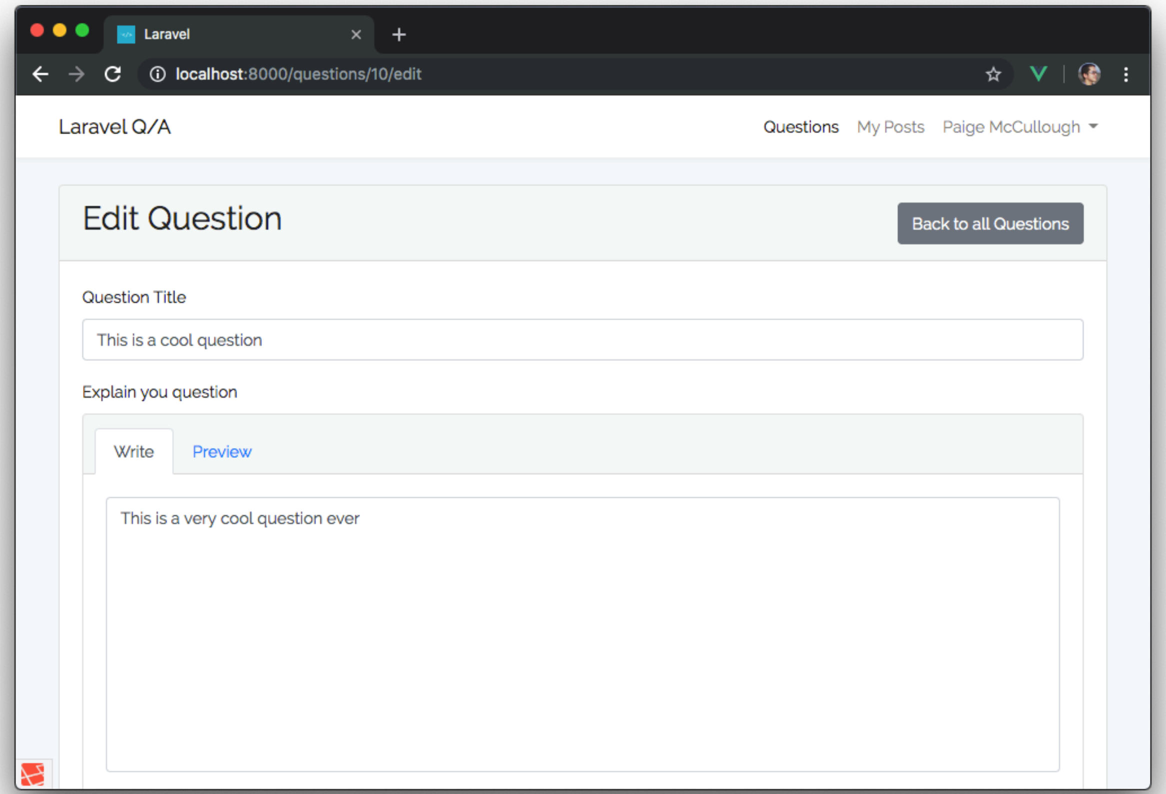
```
// ...
import EditQuestionPage from '../pages/EditQuestionPage.vue'

const routes = [
  // ...
  {
      path: '/questions/:id/edit',
      component: EditQuestionPage,
      name: 'questions.edit'
  },
  // ...
}
```

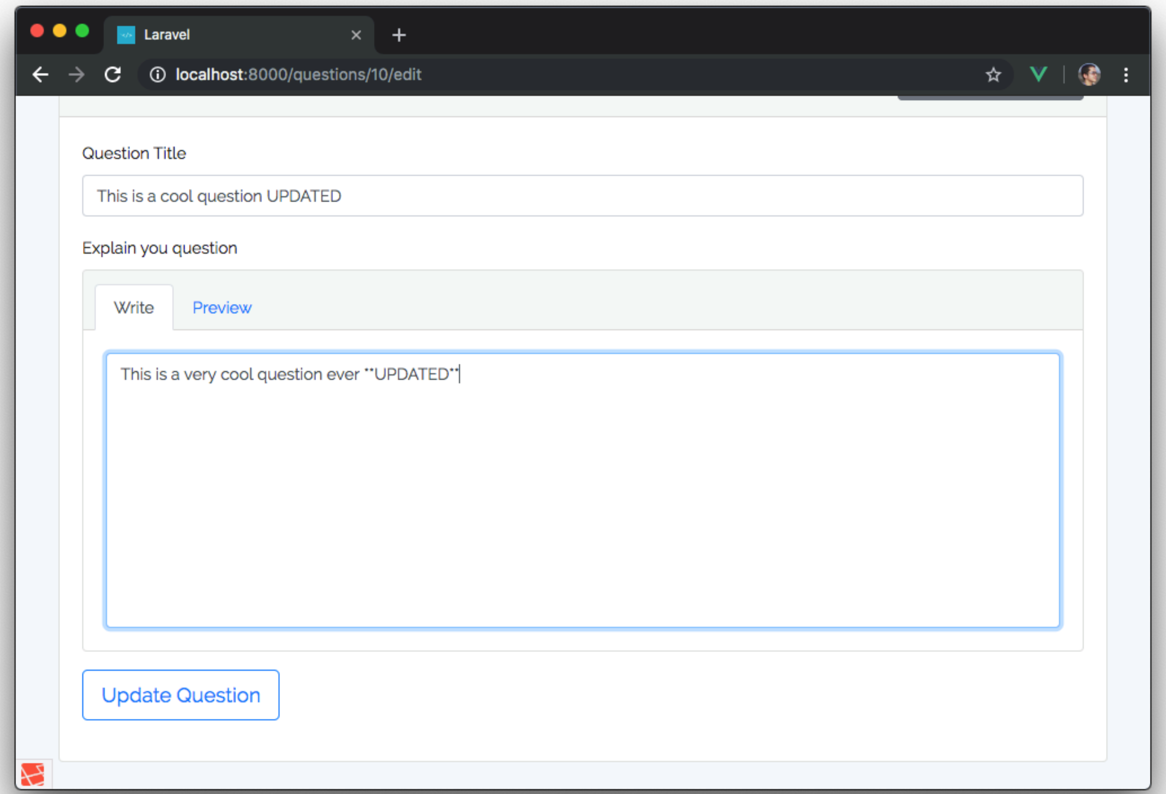
Last, let's open up the `QuestionExcerpt` component. Then replace the edit anchor (edit link) to `router-link`. Let's set the `to` attribute to point to `questions.edit`. We also need to specify `params` which contains the question id. Other attributes will be the same with the old link.

```
<router-link
    :to="{ name: 'questions.edit', params: { id: question.id } }"
    v-if="authorize('modify', question)"
    class="btn btn-sm btn-outline-info">Edit</router-link>
```

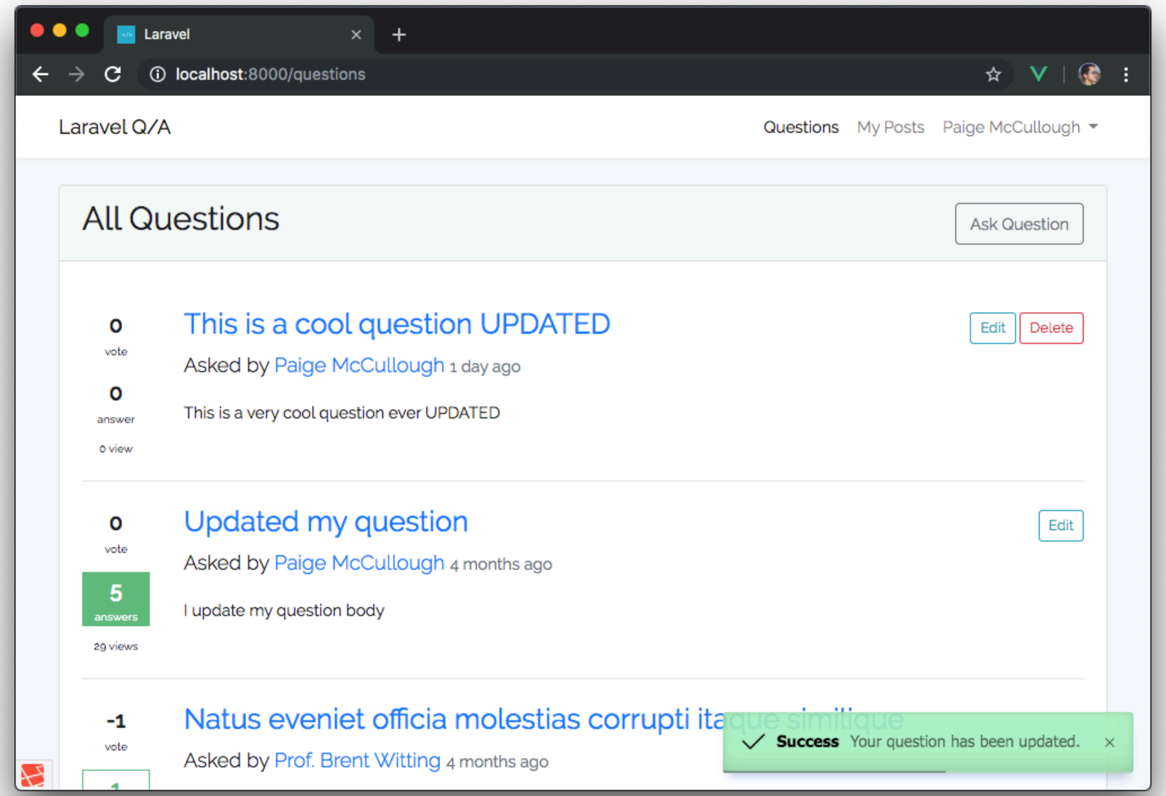
Let's save all changes and back to our browser. Now if we hit the **Edit** button. We'll have the edit question page appear. You'll also see that the old question data being populated in question `title` and `body` inputs.



Let's modify the question title and body then hit the **Update Question** button.


Laravel
localhost:8000/questions/10/edit
Question Title
This is a cool question UPDATED
Explain you question
Write
Preview
This is a very cool question ever **UPDATED**
Update Question

It will go back to all questions page. You'll see the question was updated successfully and we'll see the flash message pop up.



All Questions

Ask Question

0 vote 0 answer 0 view

This is a cool question UPDATED

Asked by Paige McCullough 1 day ago

This is a very cool question ever UPDATED

Edit Delete

0 vote 5 answers 29 views

Updated my question

Asked by Paige McCullough 4 months ago

I update my question body

Edit

-1 vote

Natus eveniet officia molestias corrupti itaque similique

Asked by Prof. Brent Witting 4 months ago

Success Your question has been updated. ×

So now we have three pages in our single page application regarding to *Question* resource: all questions page to display all questions, create question page to create a new question and edit question page to edit existing question.

In the next lesson we're going to see how to make the **Delete** button actually working. OK let's save all changes that we made today into our git repository.

```
git add .
git commit -m "Create Edit Question component"
git push origin lesson-60
```