Raffaele Senften
Inf 17f

Quicknote

Gem ‘carrierwave’

Das Gem ‘carrierwave’ wird benutzt um Bilder oder auch Dateien in Rails hochzuladen. Es ist ausserdem eines der beliebtesten Lösungen um Bilder hochzuladen.

Anwendung
Das Gem ‘carrierwave’ kann bei allen Rails Applikationen benutzt werden, bei denen man ein Bild oder auch Dateien hochladen muss. Z.B. wird dieses Gem benutzt, um ein Bild hochzuladen.

Vor- und Nachteile
+Das Gem ist sehr einfach zu installieren und anzuwenden.
- Die Konfiguration von einem «uploader» kann zu Beginn ein bisschen kompliziert sein, allerdings gibt es eine Dokumentation auf GitHub, mit der man klarkommt.

Gem ‘cloudinary’

Das Gem ‘cloudinary’ wird benutzt um Bilder und Videos in einer Cloud zu speichern.

Anwendung
Das Gem ‘cloudinary’ kann bei allen Rails Applikationen benutzt werden, bei denen man Bilder oder auch Videos in einer Cloud speichern muss.

Vor- und Nachteile
+Dank diesem Gem muss man nicht mehr selbst die Bilder und Videos speichern.
- Wenn man seine Applikation für kommerziellen Zweck verwenden will, ist cloudinary kostenpfilchtig.

Gem ‘figaro’

Das Gem ‘figaro’ wird benutz, um Daten in seiner Applikation zu verschlüsseln. Figaro nutz eine 12-Faktoren Methodik.

Anwendung
Das Gem ‘figaro’ kann bei allen Rails Applikationen benutzt werden, bei denen man seine Daten verschlüsseln will/muss.

Vor- und Nachteile
+Mit diesem Gem kann man alle seine Daten verschlüsseln.
- Nachteile gibt es meiner Meinung nach keine.

DB-Beziehung Post-Photo

Die Beziehung von Post zu Photo ist eine 1-to-many Beziehung. Dies bedeutet, dass ein Post mehrere Phototos haben kann, aber ein Photo nur zu einem Post gehört. In der Applikation wird das mit den Active Record Associations «belongs_to» und «has-many» umgesetzt.

Anwendung
Dies kann man überall brauchen, wo es eine 1:n Beziehung gibt.

Vor- und Nachteile
+Mit dieser Beziehung ist es möglich mehrere Bilder in einem Post zu posten.
- Es können Probleme entstehen, wenn man das Ganze nicht richtig konfiguriert.

Raffaele Senften
Inf 17f

Posts Controller

Rails Controller action kennt für die Anwendung von restful API sieben Methoden. Diese Methoden heissen «index, show, new, create, edit, update, destroy». Die Beziehung zwischen Post und Photo kann man im Rails Router mit Hilfe der Methoden abbilden:

```
resources :posts, only: [:index, :show, :create, :destroy] do
    resources :photos, only: [:create]
end
```

In dem Post Controller kann man die Methoden deklarieren und programmieren, was geschehen soll.

Anwendung
Dies kann man überall brauchen, wo es Beziehungen gibt.

Vor- und Nachteile
+Mit der Löschfunktion kann man automatisch die verbundenen Elemente mitlöschen.
- Nachteile gibt es meiner Meinung nach keine.

Selbstreflexion

Was habe ich gelernt?
Ich habe drei neue Gems kennengelernt. Carrierwave zum hochladen von Bildern/Dateien, cloudinary, zum Speichern von Bildern in der Cloud und figaro, um Daten zu verschlüsseln.

Wie bin ich vorgegangen beim Lernen bzw. Ausführen des Auftrages?
Wenn ich eine Aufgabe des Auftrages gelesen habe, habe ich diesen gleich ausgeführt, bevor ich an dem Auftrag weiterlas. Mit der Quicknote begann ich erst, als ich das Arbeitsblatt komplett und vollständig gelöst habe.

Was waren die Schwierigkeiten, wie konnte ich diese lösen?
Schwierigkeiten gab es bei diesem Arbeitsblatt keine. Der Auftrag war meiner Meinung nach sehr gut beschrieben.

Was habe ich nicht verstanden bzw. was konnte ich nicht lösen?
Verstanden habe ich alles sehr gut, dank der guten Beschreibung und den hilfreichen Links. Lösen konnte ich alles vollständig und auch korrekt.

Was kann ich nächstes Mal besser machen?
Leider habe ich alles auf den letzten Tag verschoben. Das nächste Mal werde ich versuchen in der Schule mehr zu lösen.

Lösungen der Aufgaben

Was bewirkt das Schlüsselwort «references»?
Das erstellt einen Foreign Key auf die gleichnamige Tabelle.

Wie ist die Namensgebung der Fremdschlüssel-Attribute?
Die Namensgebung ist <Tabellenname singular>_id. In der Applikation werden post_id und user_id erstellt.

Beziehungen mit Anpassungen für kaskadierendes Löschen
app/models/user.rb:
has_many :posts, dependent: :destroy

app/models/post.rb:

has_many :photos, dependent: :destroy

create-Methode im Post Controller – erste Zeile
Mit *@post = current_user.posts.build(post_params)* wird ein neues Post Objekt erstellt mit einem Foreign Key auf den aktuellen Benutzer und den richtigen Werten aus den post_params. Danach wird das Objekt in der @post Variabel gespeichert. Dieses Objekt wird aber noch nicht in der Datenbank gespeichert.

Rest der create-Methode
Falls der Post gespeichert werden kann wird noch über alle Images aus den Parametern iteriert und für jedes Bild ein Photo in der Datenbank erstellt mit Foreign Key auf den Post. Danach wird der Benutzer auf die Posts Seite geleitet und eine Success Nachricht «Saved …» wird im Flash mitgegeben. Falls es einen Fehler gibt wird eine Fehlermeldung («Something went wrong …») im Flash mitgegeben.

index-Methode
Hier werden die ersten 10 Posts aus der Datenbank geladen. Mit *.includes(:photos)* werden auch gleich alle Photos des Posts mitgeladen. Die Posts werden dann in der globalen Variablen *@posts* gespeichert. Ausserdem wird die globale Variable *@post* noch auf ein neues Post Objekt gesetzt.

before_action :find_post, only : [:show, :destroy]
Hier wird folgendes definiert: Bevor die show oder die destroy Methode ausgeführt wird, soll *find_post* ausgeführt werden.

Abschliessende Reflexion

Alles in allem habe ich drei neue, sehr praktische Gems kennengelernt und auch gleich einsetzten können. Dieses Arbeitsblatt ist wie die anderen auch, sehr gut beschrieben und strukturiert. Auch mit diesem Arbeitsblatt konnte ich mein Rails-Wissen erweitern.

Screenshots

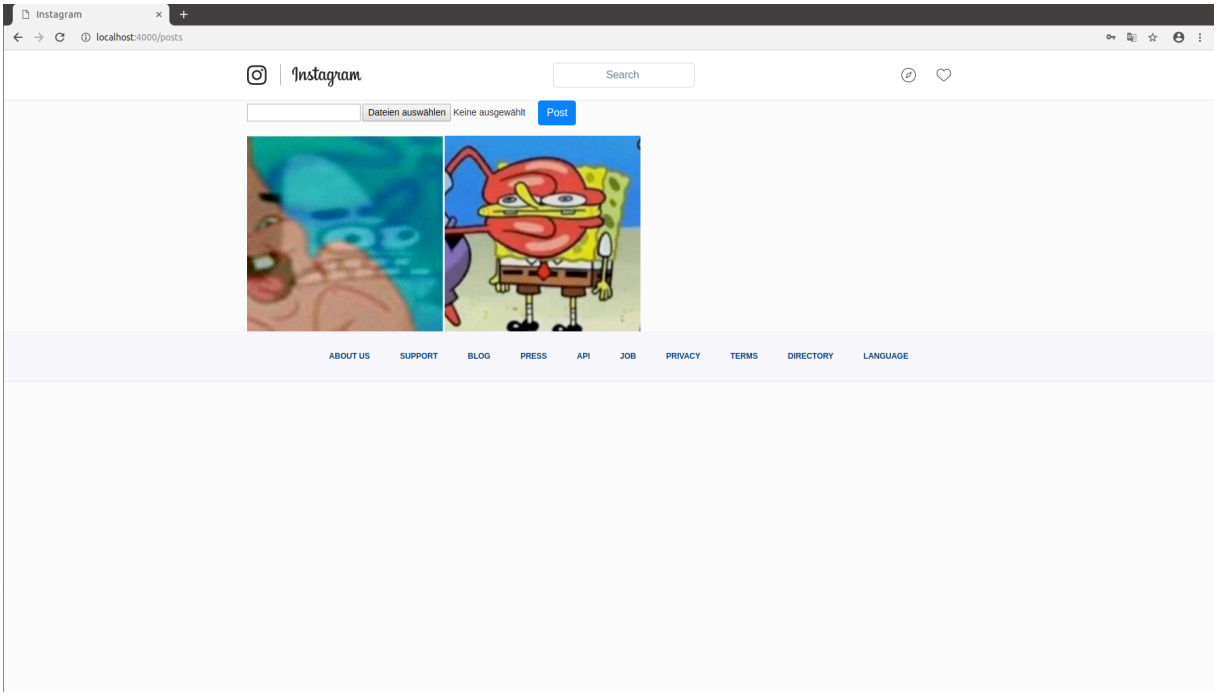

Abbildung 1: Erster Post

Ich hoffe diese Bilder sind nicht unangemessen ☺.