smartlearn
virtual learning. real success.

MODUL 151
RAILS APPLIKATION SCHRITT FUER SCHRITT

Michael Abplanalp

Rails Applikation Schritt für Schritt

Dieses Dokument soll helfen, eine Rails-Applikation Schritt für Schritt zu erstellen. Es wird eine neue Applikation erstellt: Es werden zuerst Musikalben und dann einzelne Songs erfasst. Jeder Song wird genau einem Album zugewiesen.

Es handelt sich teilweise um eine Wiederholung des Schlussteils von Modul 133. Die Arbeit soll als Repetition und Festigung des Rails MVC-Frameworks angesehen werden.

Die einzelnen Aufgaben:

- Models *Album* und *Song* erstellen, Tabellen anpassen und mit `rails db:migrate` in die DB schreiben.
- Ein paar Alben und Songs in der Rails Konsole erfassen.
- Controller *Albums* und *Song*s erstellen.
- Routen erstellen: Wir verwenden `resources`, weil damit die sieben notwendigen Routen plus Helpers automatisch erstellt werden.
- Notwendige Methoden in den beiden Controllern *Albums* und *Songs* erfassen und entsprechende Views erstellen. Die Funktionen:
  - Alben bzw. Songs in einer Liste anzeigen
  - Album bzw. Fachgebiet in einem Formular erfassen
  - Album bzw. Fachgebiet in die DB speichern
  - Album bzw. Song in einem Formular ändern
  - Änderungen in die DB speichern
  - Album bzw. Song löschen

  Zusatzfunktionen:
  - Anzahl Songs pro Album ermitteln und anzeigen
  - Laufzeit pro Album berechnen
  - Anzahl der Alben anzeigen
  - Songnummer ermitteln
  - Songs pro Album anzeigen

Model erstellen

Zu Beginn einer Applikation wird stets die Datenmodellierung gemacht. Wir haben Alben und Songs, die Beziehung ist 1:m (jeder Song ist genau einem Album zugeordnet, jedes Album kann 0, 1 oder mehrere Songs enthalten).

1. Jede Tabelle wird als eigenes Model in der Konsole erstellt:

```
$ rails g model Album
$ rails g model Song
```

Der Name des Models steht in der Einzahl.

Es werden dabei u.a. folgende Dateien erstellt:

```
app/db/migrate/20180430133029_create_albums.rb
app/db/migrate/20180430133034_create_songs.rb
app/models/album.rb
app/models/song.rb
```

2. Als nächstes definieren wir die Tabellenstrukturen:

`app/db/migrate/20190125085411_create_albums.rb`

```
class CreateAlbums < ActiveRecord::Migration[5.2]
  def change
    create_table :albums do |t|
      t.string :band
      t.string :title
      t.integer :year
      t.integer :no_songs
      t.integer :duration
      t.timestamps
    end
  end
end
```

Die Klasse wurde mit dem Model automatisch erstellt, wir müssen nur noch die Tabellenattribute einfügen: *band, title, year, no_songs* (Anzahl Songs) und *duration* (Dauer des Albums in Sekunden) ein.

In Rails läuft Vieles automatisch ab. So wird der Primärschlüssel mit dem Namen `id` von Rails automatisch erstellt und auch verwaltet. Sie brauchen sich also nicht darum zu kümmern.

Mit `t.timestamps` werden die beiden Attributen *created_at* und *updated_at* erstellt. Auch diese bewirtschaftet Rails automatisch, bei Bedarf können Sie diese verwenden (z.B. anzeigen).

`app/db/migrate/20190125085418_create_songs.rb`

```
class CreateSongs < ActiveRecord::Migration[5.2]
  def change
    create_table :songs do |t|
      t.integer :album_id
      t.integer :position
      t.string :title
      t.integer :duration
      t.timestamps
    end
  end
end
```

Wir fügen die vier Attribute *album_id* (Fremdschlüssel Album), *position* (`Nummer des Songs`), *title* und *duration* (Dauer des Albums in Sekunden) ein.

3. Wir definieren die Beziehung zwischen den Tabellen und die Validationen:

`app/models/album.rb`

```
class Album < ApplicationRecord
  has_many :songs
  validates_presence_of :band
  validates_presence_of :title
end
```

Ein Album kann 0, 1 oder mehrere Songs haben. Die Attribute *band* und *title* sind obligatorisch, die anderen Attribute nicht.

`app/models/song.rb`

```
class Song < ApplicationRecord
  belongs_to :album
  validates_presence_of :title
end
```

Ein Song muss immer genau einem Album zugeordnet sein. Das Attribut *title* ist obligatorisch, die anderen Attribute sind es nicht.

Beachten Sie die Schreibweisen in Ein- und Mehrzahl:

- Das Model wurde mit Einzahl erstellt: *Album* und *Song*
- Beim Erstellen der Tabellen bildete Rails aus den Modelnamen die Mehrzahl für die Tabellennamen (*albums* und *books*).
- Die Beziehung has_many :songs steht in der Mehrzahl, weil ein Album mehrere Songs haben kann
- Die Beziehung belongs_to :album steht in der Einzahl, weil ein Song immer genau einem Album zugeordnet ist

Neben den Beziehungen können wir Validationen definieren: Titel und Band bei den Alben und Titel bei den Songs sind obligatorische Attribute. Ohne die obligatorischen Attribute kann kein Datensatz gespeichert werden.

4. Zuletzt müssen wir die Datenbank migrieren, d.h. alle anstehenden Änderungen durchführen (in der Konsole):

```
$ rails db:migrate
```

Alternative: Migration erstellen

Mit Rails Migrations können Anpassungen an Datenstrukturen strukturiert und organisiert vorgenommen werden. Informationen, wie Sie mit Migrations arbeiten, finden Sie in Teil 4 des Moduls 133.

Daten bearbeiten

Für die Entwicklung einer Applikation ist es meist von Vorteil, wenn bereits Daten vorhanden sind. Daten können einfach in der Rails Konsole erfasst und bearbeitet werden.

```
$ rails c
2.5.1 :001 > Album.connection
```

Zuerst muss mit Album.connection oder Song.connection eine Verbindung zum Model erstellt werden.

Daten erfassen

Erfassen Sie mind. 3 (eigene!) Alben in der Konsole:

```
> Album.create(band: "Robyn", title: "Honey", year: 2018)
> Album.create(band: "Gorillaz", title: "The Now Now", year: 2017)
> Album.create(band: "Neuschnee", title: "Okay", year: 2018) >
```

Album bezieht sich auf das Model.

Erfassen Sie mind. 2 Songs pro Album in der Konsole. Mit *duration* geben Sie die Lauflänge des Songs in Sekunden an.

```
> Song.create(album_id: 1, title: "Missing U", duration: 297)
> Song.create(album_id: 1, title: "Human Being", duration: 226)
> Song.create(album_id: 2, title: "Humility", duration: 198)
> Song.create(album_id: 2, title: "Tranz", duration: 163)
> Song.create(album_id: 3, title: "Der Zeitgeist macht buh",
duration: 204)
> Song.create(album_id: 3, title: "It's Ok To Feel Lost", duration:
278)
```

Song bezieht sich auf das Model.

`album_id` ist der Fremdschlüssel auf das Fachgebiet. Achten Sie darauf, nur gültige Werte zu verwenden.

Alternative zum Erfassen von Daten:

```
> s = Song.new
> s.album_id = 1
> s.title = "Missing U"
> s.duration = 297
> s.save
```

Daten ändern

Statements, die nur 1 Datensatz zurückgeben:

```
> s = Song.first
> s.title = "Missing You"
> s.save
```

Statements, die 0, 1 oder mehrere Datensätze zurückgeben:

```
> s = Song.where(title: "Missing You")
> s[0].title = "Missing U"
> s.save
```

Es wird stets ein Array zurückgegeben, auch wenn nur 1 Datensatz gefunden wird! D.h. wir müssen immer über den Index auf einen Datensatz zugreifen: Der Index beginnt bei 0, d.h. [0] ist der erste zurückgegebene Datensatz.

Daten löschen

```
> Song.last.destroy
oder
> Song.last.delete
```

Controller erstellen

Auch wenn im Modul 133 z.T. ein Controller für mehr als ein Model verwendet wird, ist das nicht die Norm und auch nicht zu empfehlen. Die Vorteile von einem Controller pro Model wird später in diesem Arbeitsblatt ersichtlich.

```
$ rails g controller Albums
$ rails g controller Songs
```

Der Name des Controllers steht in der Mehrzahl.

Es werden dabei u.a. folgende Dateien und Verzeichnisse erstellt:
`app/controllers/albums_controller.rb`
`app/controllers/songs_controller.rb`
`app/views/albums`
`app/views/songs`

Routen erstellen

Rails hat ein REST API, d.h. alle Seiten und Funktionen einer Applikation werden via URL aufgerufen und ausgeführt. Der Rails Router erkennt URLs und führt die entsprechende Aktion (Methode) im Controller aus.

Die Routen und das Mapping sind in der Datei `config/routes.rb` festgehalten. Wir fügen die folgenden zwei Einträge in diese Datei ein:

```
Rails.application.routes.draw do
  resources :albums
  resources :songs
end
```

Damit erstellt Rails automatisch sieben Routen mit entsprechenden Mappings zum Controller. Für den Albums-Controller sind das die folgenden Einträge:

HTTP Verb	Pfad	Controller#Action	Beschreibung
GET	/albums	albums#index	Liste mit allen Alben anzeigen
GET	/albums/new	albums#new	HTML-Formular anzeigen zum Erfassen eines neuen Albums
POST	/albums	albums#create	Neues Album in die DB speichern
GET	/albums/:id	albums#show	Details eines Albums anzeigen
GET	/albums/:id/edit	albums#edit	HTML-Formular anzeigen zum Bearbeiten eines Albums
PATCH/PUT	/albums/:id	albums#update	Änderungen in die DB speichern
DELETE	/albums/:id	albums#destroy	Ein Album löschen

Entsprechend werden natürlich auch die Routen für den Songs-Controller generiert.

Für jede Route muss im Controller eine entsprechende Methode eingetragen werden, also die Methoden: *index, new, create, show, edit, update, destroy*

Weitere Routen können in die Datei eingefügt werden, z.B. zum Erstellen einer Liste mit allen Bands:

```
get 'albums/bands_index', to: 'albums#bands_index'
```

Auch hier muss natürlich die Methode *bands_index* im Controller vorhanden sein.

Neben den sieben Routen werden ein paar "Helpers" (Programmierhilfen) zur Verfügung gestellt, damit die Routen einfacher erstellt werden können:

Helper	Gibt zurück
albums_path	/albums
new_album_path	/albums/new
edit_album_path(:id)	/albums/:id/edit
album_path(:id)	/albums/:id

Die Routen und Helpers werden in den Views verwendet, um die verschiedenen CRUD-Operationen auszuführen.

Beispiel:

```
<table
  <% Album.all.each do |a| %>
    <td><%= a.band</td>
    <td><%= a.title</td>
    <td><%= link_to "Ändern", edit_album_path(a.id) %></td>
    <td><%= link_to "Löschen", album_path(a.id), method: :delete%></td>
  <% end %>
</table>
```

Der Ändern-Link wird in HTML wie folgt ausgegeben:
`<a href="/albums/1/edit">Ändern</a>`, wobei 1 = die ID des ersten Albums

Der Löschen-Link wird in HTML wie folgt ausgegeben:
`<a href="/albums/2">Löschen</a>`, wobei 2 = die ID des zweiten Albums

Controller und Views für die Alben erstellen

In einem ersten Schritt erstellen wir den Controller und die Views für die Alben. Denn die Alben sind unabhängig von den Songs und müssen vor den Songs erfasst werden. Der komplett notwendige Code ist im Folgenden aufgeführt. In einem zweiten Schritt werden Sie den Controller und die Views für die Songs selbst erarbeiten.

Methoden im Controller erfassen

Der Controller ist bereits vorhanden, Sie müssen nun die notwendigen Methoden erfassen.

`app/controllers/ albums_controller.rb`:

```
class AlbumsController < ApplicationController
  def index
    @albums = Album.all
  end

  def new
  end

  def create
    album = Album.new(album_params)
    if album.save
      redirect_to action: "index"
    else
      render action: "new"
    end
  end

  def edit
    @album = Album.find(params[:id])
  end
```

```
  def update
    album = Album.find(params[:id])
    if album.update_attributes()
      redirect_to action: "index"
    else
      render action: "edit"
    end
  end

  def destroy
    if Song.where(album_id: params[:id]).count > 0
      redirect_to action: "index"
    else
      Album.find(params[:id]).destroy
      redirect_to action: "index"
    end
  end

  def album_params
    params.require(:album).permit(:band, :title, :year)
  end
end
```

Views erstellen

Schliesslich müssen noch die Web-Seiten erstellt werden. Es braucht nicht für alle Aktionen eine eigene Seite:

- *index* zeigt die Liste mit allen Fachgebieten
- *new* führt zum Formular, in dem ein neues Album erfasst werden kann
- *create* speichert das Album in die DB und ruft danach den Index auf (d.h. braucht keine eigene View)
- *show* zeigt die Details eines Albums an (das lassen wir vorläufig weg)
- *edit* führt zum Formular, in dem ein Album bearbeitet werden kann
- *update* speichert die Änderungen in die DB und ruft danach den Index auf
- *destroy* löscht auf der Indexseite ein Album und ruft den Index erneut auf

Wir müssen für die 3 Views je eine Datei im Verzeichnis `app/views/albums` erstellen: `index.html.erb, new.html.erb` und `edit.html.erb`

albums#index

Vorläufig wollen wir die Attribute *Band, Titel* und *Jahr* anzeigen. Es werden Links zum Ändern und Löschen von Alben und zum Erfassen eines neuen Albums erstellt.

Alben

Band	Titel	Jahr		
Robyn	Honey	2018	Ändern	Löschen
Gorillaz	The Now Now	2018	Ändern	Löschen
Neuschnee	Okay	2018	Ändern	Löschen

Album erfassen
Songs

app/views/albums/index.html.erb:

```
<h3>Alben</h3>
<table>
  <tr>
    <th>Band</th>
    <th>Titel</th>
    <th>Jahr</th>
    <th colspan="2"> </th>
  </tr>
  <% @albums.each do |a| %>
    <tr>
      <td><%= a.band %></td>
      <td><%= a.title %></td>
      <td><%= a.year %></td>
      <td><%= link_to "Ändern", edit_album_path(a.id) %></td>
      <td><%= link_to "Löschen", album_path(a.id), method: :delete %></td>
    </tr>
  <% end %>
</table>
<%= link_to "Album erfassen", new_album_path %><br />
<%= link_to "Songs", songs_path %>
```

In diesem Dokument werden HTML- und CSS-Code auf ein Minimum beschränkt. Auf die Formatierung der Seiten wird nicht näher eingegangen. Das Design der Applikation, in der die Screenshots erstellt worden sind, wurde mit Bootstrap 4 erstellt.

Informationen zu Bootstrap finden Sie in den Modulunterlagen des Moduls 133 oder unter https://getbootstrap.com/ und https://www.w3schools.com/bootstrap4/

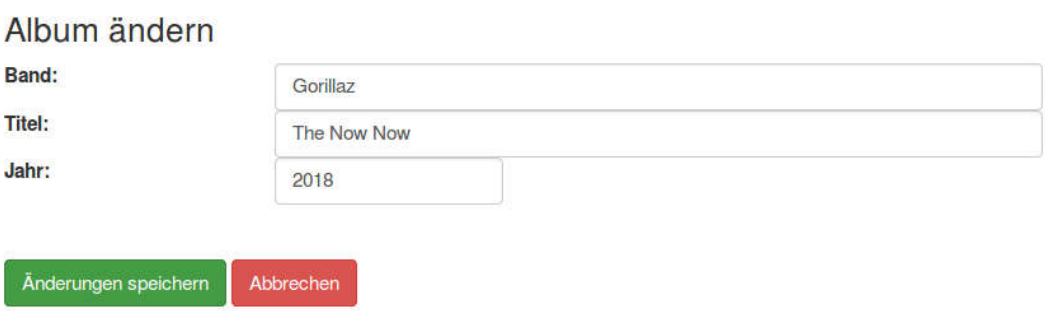
app/views/albums/new.html.erb:

```
<h3>Album erfassen</h3>
<%= form_for :album, url: {action: "create"} do |f| %>
  <%= f.label :band, "Band:" %>
  <%= f.text_field :band %><br />
  <%= f.label :title, "Titel:" %>
  <%= f.text_field :title %><br />
  <%= f.label :year, "Jahr:" %>
  <%= f.text_field :year %><br />
  <%= f.submit "Album speichern" %>
```

```
  <%= link_to "Abbrechen", albums_path %>
<% end %>
```

Testen Sie die Funktion, indem Sie zwei neue Alben erfassen (eins davon löschen wir dann gleich wieder).

albums#edit



`app/views/albums/edit.html.erb`:

```
<h3>Album bearbeiten</h3>
<%= form_for @album, url: {action: "update"} do |f| %>
  <%= f.label :band, "Band:" %>
  <%= f.text_field :band %><br />
  <%= f.label :title, "Titel:" %>
  <%= f.text_field :title %><br />
  <%= f.label :year, "Jahr:" %>
  <%= f.text_field :year %><br />
  <%= f.submit "Änderungen speichern" %>
  <%= link_to "Abbrechen", albums_path %>
```

Testen Sie die Funktion, indem Sie ein Album auswählen und Änderungen durchführen.

albums#destroy

Versuchen Sie zuerst, ein Album zu löschen, bei dem Sie bereits Songs erfasst haben. Es sollte keinen Programmabsturz geben, sondern die Indexseite sollte einfach neu geladen werden (ohne Fehlermeldung).

Versuchen Sie anschliessend, ein soeben erfasstes Album zu löschen (das noch keine Songs hat).

Controller und Views für die Songs erstellen

Nun ist es an der Zeit, dass Sie selbst und ohne Vorlage etwas erarbeiten: Die Methoden des Songs-Controller und die Views für die Songs.

Controller

Es müssen dieselben Methoden erstellt werden wie bei den Alben:
index, new, create, show, edit, update, destroy

Grundsätzlich ist der Code derselbe wie bei den Alben, mit folgenden Abweichungen:

- Das Model, die Variablennamen und die Parameterliste müssen von "Album" zu "Song" geändert werden
- Beim Erfassen und Ändern eines Songs werden alle Alben in einer Liste angezeigt. D.h. in den Methoden *new* und *edit* müssen wir alle Alben in einer Instanzvariablen speichern.
- Die Überprüfung, ob ein Album gelöscht werden kann, fällt weg. Denn die Songs sind unabhängig von den Alben.

Views

Auch die Views können zu grossen Teilen von den Alben übernommen werden.

songs#index

Songs

Band	Album	Titel	Dauer		
Robyn	Honey	Missing U	397	Ändern	Löschen
Robyn	Honey	Human Being	226	Ändern	Löschen
Gorillaz	The Now Now	Humility	198	Ändern	Löschen
Gorillaz	The Now Now	Tranz	163	Ändern	Löschen
Neuschnee	Okay	Der Zeitgeist macht buh	204	Ändern	Löschen
Neuschnee	Okay	It's Ok To Feel Lost	278	Ändern	Löschen

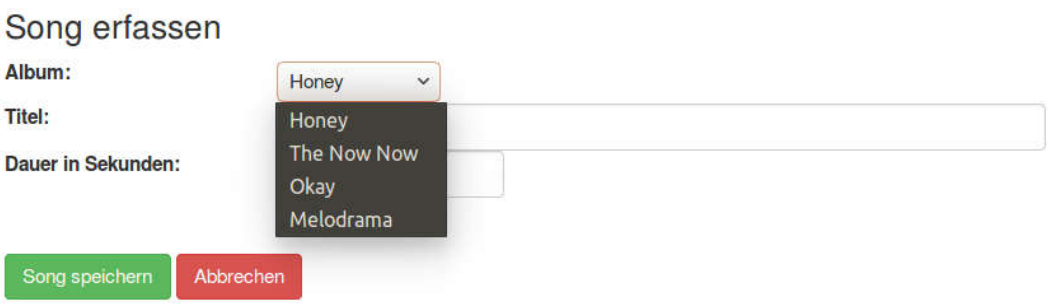
Song erfassen
Alben

Wir wollen im Index nicht nur die Songinformationen anzeigen, sondern auch das Album (plus Band), zu dem der Song gehört. Dazu holen wir im Controller wie üblich die Songs mit `Song.All` und können in der View mit `s.album.band` und `s.album.title` auf die Albuminformationen zugreifen.

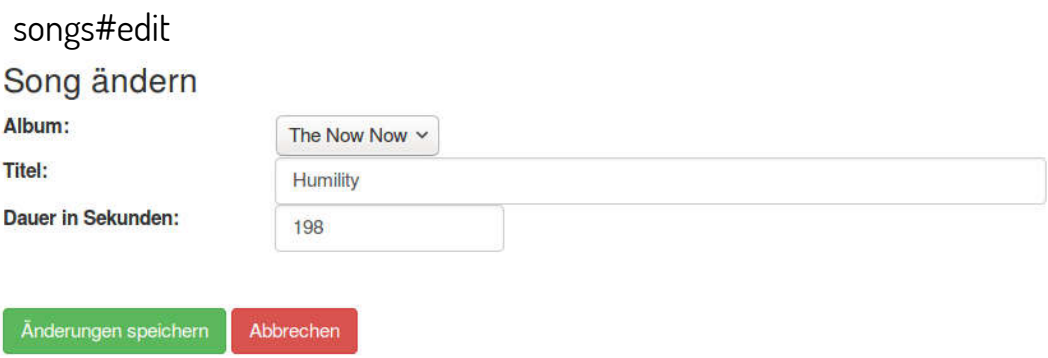
In diesem Formular haben wir ein zusätzliches Element: Ein Drop-Down-Liste mit allen Alben. Wir erzeugen die Liste mit
`f.collection_select :album_id, @albums, :id, :title`:

- *f* ist der Verweis auf das Formular
- *collection_select* erzeugt eine HTML-Drop-Down-Liste (<select><option>…)
- *album_id* ist das Attribut im Model, in dem der Wert gespeichert wird (der Fremdschlüssel auf das Album)
- *@albums* ist das Objekt mit allen Alben
- *id* ist der Wert, der einem Listeneintrag hinterlegt wird (Primärschlüssel des Albums)
- *title* ist das Attribut des Albums, das in der Drop-Down-Liste angezeigt wird

Mit geöffneter Drop-Down-Liste sieht das Formular wie folgt aus:



Testen Sie die Funktion, indem Sie ein paar Songs erfassen.



Die Drop-Down-Liste wird genau gleich erzeugt wie bei `song#new`.

Testen Sie die Funktion, indem Sie einen Song auswählen und Änderungen durchführen.

songs#destroy

Testen Sie die Funktion, indem Sie aus der Liste der Songs (Index) einen Song löschen.

Zusatzaufgaben

Dieses Kapitel beschreibt die Zusatzaufgaben. Anders als in den vorherigen Kapiteln müssen Sie die Lösung selbst finden. Es gibt nur Hinweise darauf, wo etwas angepasst bzw. ergänzt werden muss.

Wenn Sie nicht weiterkommen, konsultieren Sie das Internet, den Banknachbarn oder die Lehrperson.

albums#index erweitern

Die Seite soll wie folgt aussehen:

Aufgaben:

- Die zwei neuen Spalten *Dauer* und *Songs* sollen eingefügt werden.
- *Dauer* ist die Summer der Dauer von allen Songs in Sekunden.
  Die Dauer des Albums wird in einem Attribut gespeichert. Da die Daten des Models auf verschiedene Weise bearbeitet werden können (via App im Browser, in der Konsole, usw.), ist es am besten, wenn die Dauer jedes Mal neu berechnet wird, wenn der Index aufgerufen wird. D.h. die Berechnung und das Speichern in die Datenbank erfolgt im Albums-Controller.
- *Songs* ist die Anzahl Songs des jeweiligen Albums.
  Was für die Dauer gilt, gilt auch für die Anzahl Songs: Die Berechnung sollte im Albums-Controller (Methode *index*) erfolgen.
- Der Albumtitel ist ein Link auf `albums#show`. Die View wird in der nächsten Zusatzaufgabe beschrieben.
- *Anzahl Alben* ist die Anzahl der in der Datenbank erfassten Alben.
  Dieser Wert wird nicht in der Datenbank festgehalten, sondern bei jedem Anzeigen der Seite neu berechnet (Methode *index* des Albums-Controller).

albums#show erstellen

Wenn im Albumindex auf den Albumtitel geklickt wird, soll folgende Seite angezeigt werden:

Lorde "Melodrama" (2017)

Nr.	Titel	Dauer
1	Green Light	234
2	Sober	197
3	Homemade Dynamite	189

Zurück

Aufgaben:

- Im Seitentitel werden *Band*, *Albumtitel* und *Jahr* angezeigt (die Daten sind im Albums-Controller und nicht etwa in der View zu holen).
- In der Liste werden alle Songs des ausgewählten Albums angezeigt.
- Die Nummer *Nr* ist die Positionsnummer des Songs innerhalb des Albums.
  Die Position wird beim Aufrufen der Seite ermittelt und im Attribut *position* in die Datenbank gespeichert (wird wiederum im Albums-Controller erledigt).

songs#index erweitern

Die Seite soll wie folgt aussehen:

Songs

Band	Album	Nr.	Titel	Dauer		
Robyn	Honey	1	Missing U	397	Ändern	Löschen
Robyn	Honey	2	Human Being	226	Ändern	Löschen
Gorillaz	The Now Now	1	Humility	198	Ändern	Löschen
Gorillaz	The Now Now	2	Tranz	163	Ändern	Löschen
Neuschnee	Okay	1	Der Zeitgeist macht buh	204	Ändern	Löschen
Neuschnee	Okay	2	It's Ok To Feel Lost	278	Ändern	Löschen
Lorde	Melodrama	1	Green Light	234	Ändern	Löschen
Lorde	Melodrama	2	Sober	197	Ändern	Löschen
Lorde	Melodrama	3	Homemade Dynamite	189	Ändern	Löschen

Song erfassen
Alben

Die Seite wird mit der Songnummer ergänzt.