Web-basierte Anwendungen

Praktikumsaufgabe 7: Ein eigenes Rails-Projekt

Studiengänge AI (4140) & WI (4120)

Projekt: Fachliche Ziele

- Ziel: Mini-Bibliothek verwalten
  - Verwaltung einer „überschaubaren“ Menge von Büchern
    - Geringe Zahl der Bücher (ca. 10…100) ermöglicht Vereinfachungen im Vergleich zu realen Bibliotheksbeständen
  - Bücher können in mehreren Exemplaren im Bestand vorliegen
  - Exemplare besitzen eine Bestandsnummer, Bücher eine ISBN
  - Exemplare besitzen eine Regal-Signatur
  - Bücher können mehrere Autor:innen aufweisen, diese können mehrere Bücher verfasst haben
  - Benutzer können Buch-Exemplare leihen und zur Ausleihe vormerken
- Übergeordnetes Ziel
  - Erstmals alle Komponenten einer webbasierten Anwendungen im Zusammenhang erleben und selbst zusammenfügen – incl. Erforderlicher Recherchen
- Organisatorisches
  - Diese Beschreibung ist der erste von zwei Teilen zum Projekt
  - In diesem Teil stehen die Modelle und Kernfunktionen im Vordergrund. Teil 2 widmet sich Verfeinerungen wie Gestaltung, Bedienbarbeit, Interaktivität, Tests
  - Aus Zeitgründen wird Teil 2 in diesem Semester nicht ausgeführt/bewertet

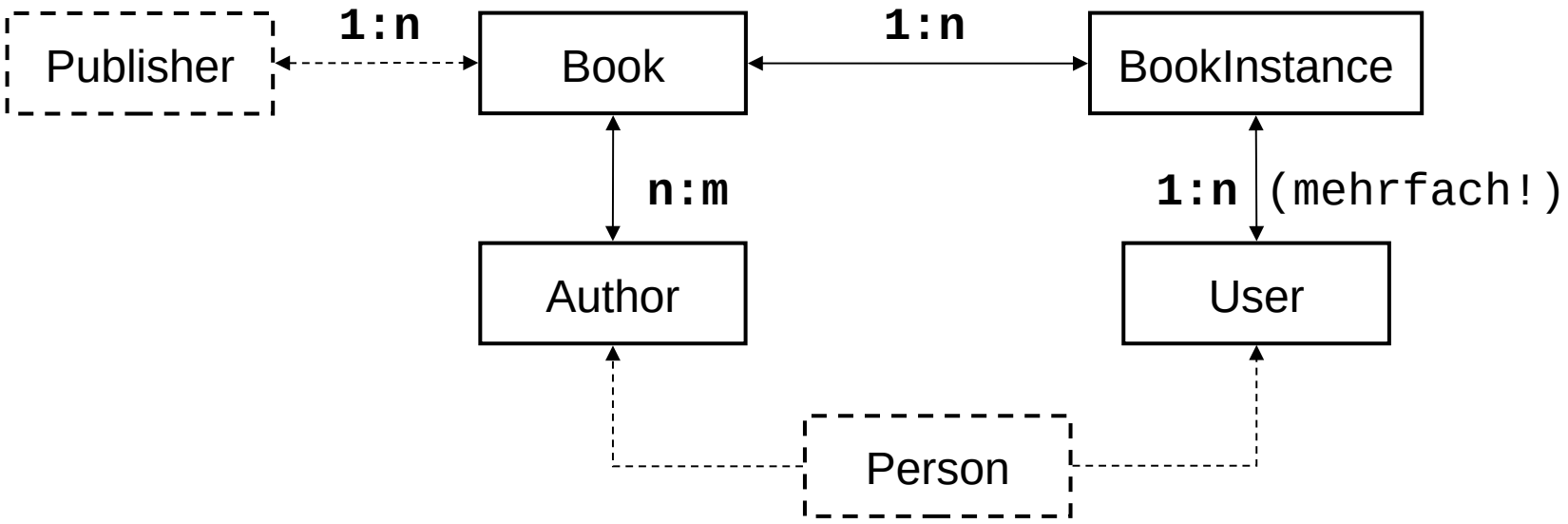
Projekt: Modelle

- Modelle
  - **Author**
    - Erfasst die Stammdaten eines Buch-Autors – hier nur angedeutet
  - **Book**
    - Zur Verwaltung der Stammdaten eines Buchs
  - **BookInstance**
    - Diese Objekte repräsentieren die Bibliotheksbestände
    - Es kann mehrere Exemplare eines Buches im Bestand geben
    - Ausleihen & Vormerkungen werden über diese Objekte verwaltet
  - **User**
    - Erfasst Benutzerdaten, auch Sperrvermerk (darf nicht ausleihen)

Projekt: Modelle

- Modell-Vereinfachungen
  - **Publisher**
    - Statt Verlage als eigene Modelle zu erfassen und Relationen mit Büchern herzustellen, vereinfachen wir hier und geben den Verlag eines Buchs nur per String ein.
  - **Person**
    - Author und User lassen sich als Unterklassen einer Basisklasse „Person“ modellieren. Dies ist mit Rails möglich, verkompliziert aber das Projekt.
    - Zur Vereinfachung nehmen wir daher hier Redundanz in Kauf und modellieren „Author“ und „User“ unabhängig voneinander
      - Zur Vererbung innerhalb von Models:
        „*Single Table Inheritance*“ recherchieren, wer hier mehr wissen will.
      - Tipp: Gemeinsame Methoden lassen sich als “concerns” an einer Stelle verwalten.

Projekt: Modelle und ihre Relationen



DB-Tabelle: books

- Spalte
  - title
  - publisher
  - pub_year
  - edition
  - isbn
- Datentyp
  - string
  - string
  - integer
  - integer
  - string (ISBN-10 kann “X” enthalten)

- Anforderungen
  - pub_year:
    Default: Aktuelles Jahr
  - edition:
    Default: 1
  - isbn: Muss-Feld
    13 Ziffern oder 10 Ziffern mit Trenn-zeichen ‚-‘ bzw. mit ‘X’ anstelle der letzten Ziffer (Prüfziffer)

DB-Tabelle: book_instances

- Spalte
  - book_id
  - shelfmark
  - purchased_at
  - lended_by_id
  - reserved_by_id
  - checkout_at
  - due_at
  - returned_at
- Datentyp
  - integer
  - string
  - date
  - integer
  - integer
  - datetime
  - date
  - datetime
- Fremdschlüssel
  - book_id → Book
  - lended_by_id → User
  - reserved_by_id → User
- Tipp
  - Fremdschlüssel realisieren z.B. mit book:references statt mit book_id:integer

DB-Tabelle: users

Spalte	Datentyp
family_name	string
given_name	string
address	text
email	string
password	string
blocked	boolean
remarks	text

- Hinweise
  - blocked: Kennzeichen, dass diese Person nicht (mehr) ausleihen darf. Default: false
  - remarks: Platz für eine Begründung für die verhängte Sperre

DB-Tabelle: authors

- Spalte
  - family_name
  - given_name
  - affiliation
- Datentyp
  - string
  - string
  - text

- Hinweise:
  - „affiliation“ benennt z.B. die Hochschule oder Firma, bei der der Autor / die Autorin arbeitet
  - Wir verzichten bewusst auf weitere Details wie Anschrift, E-Mail, Lebenslauf ...

Projekt: Modelle

- Vorgehen
  - Erzeugen Sie zu jedem der 4 Modelle ein vollständiges Gerüst für CRUD-Operationen, typischerweise mit:

    ```
    $ rails generate scaffold Book title:string …
    ```

  - Wechseln Sie in Ordner **db/migrate**
  - Bearbeiten Sie die dort angelegten Dateien, um die Default-Anforderungen zu erfüllen.
  - In der Datei zu book_instances ändern:

    von: `t.references :lended_by, null: false, foreign_key: true`

    in: `t.references :lended_by, foreign_key: {to_table: :users}`
  - Ebenfalls die Zeile zu `t.references :reserved_by` gleichartig ändern!
  - Erzeugen Sie für die n:m-Beziehung zwischen Author und Book eine Join-Tabelle „authors_books“ (ohne id-Feld !)
    - Tipp: **`rails generate migration authors_books`**

      dann die erzeugte Datei manuell befüllen
    - Anleitung: https://guides.rubyonrails.org/association_basics.html, insb. Abschnitt 3.3.2 „Creating Join Tables“ sowie der Anhang zu diesen Folien
  - Erst nach Abschluss aller Migration-Änderungen ausführen:

    ```
    $ rake db:migrate
    ```

Projekt: Modelle

- Vorgehen (Forts.)
  - Wechseln Sie in **`app/models`**
  - Bearbeiten Sie die dort angelegten Dateien, um die Assoziationen (*has_many, has_one, belongs_to, has_and_belongs_to_many*) zu ergänzen bzw. zu korrigieren
    - Anleitung: https://guides.rubyonrails.org/association_basics.html
    - Hinweis: Die beiden Assoziationen zwischen User und BookInstance erfordern eigenes Nachdenken und Recherchieren!
  - Beachten Sie dabei die Integrität der Relationen: Löschen eines Objekts, auf das verwiesen wird, muss Folgen für die Verweise haben
    - BookInstance: unkritisch – enthält nur Verweise, aber ist nicht Ziel von Verweisen. Kann einfach gelöscht werden.
    - Book: Abhängige BookInstance-Objekte ebenfalls löschen lassen!
    - Author: Rails sollte die Join-Tabelle automatisch aktualisieren, also diese Person aus den Autorenlisten der betroffenen Bücher streichen.
    - User: Sonder-Anforderungen
      - Löschen verhindern, falls Benutzer:in noch Bücher geliehen hat
      - Eventuelle Reservierungen zurücksetzen!
  - Hinweis: Sicherstellung dieser Integritäten ist teils einfach, teils aber auch nicht. Tipp: Zunächst zurückstellen und später einbauen (aber nicht vergessen).

Projekt: Modelle

- Vorgehen (Forts.)
  - Validierungen
    - Ergänzen Sie die Modelle um sinnvolle Validierungen, wie z.T. bereits in der Vorbereitung (Übung 04) geschehen
    - Ermessen Sie selbst, welche Angaben vorhanden bzw. korrekt formatiert anzugeben sind, damit die gewünschten Prozesse funktionieren.
    - Fall User: Verhindern Sie Ausleihen im Fall “blocked” per Validierung und mit einer passenden Fehlermeldung. Bauen Sie dazu eine eigene Validierungsfunktion.

Projekt: Erweiterungen

- Erweiterungen
  - Vorbelegungen/Aktualisierungen (per Callback-Technik)
    - Author, User: keine
    - Book: pub_year (DB-Vorgabe auf aktuelles Jahr ändern), nur für Action „new“
    - BookInstance:
      - purchased_at: aktuelles Datum („initialize“),
      - due_at: leer (ohne Ausleihe) oder „checkout_at“ + 4 Wochen („before_save“)
    - HINWEIS: Bei knapper Zeit zurückstellen
  - Hilfsmethoden
    - Author, User: Methode „short_name“, ergibt z.B. „H. Werntges“
    - ~~User: Methode „authenticate“ (s.o.)~~
      (Gehört zu Teil 2, der in diesem Semester nicht zur Ausführung kommt)
    - Alle: Methode „custom_select“ (siehe “Filter”)
    - Ggf. weitere Methoden nach eigenem Ermessen

Projekt: Views

- Layout
  - OPTIONAL: Stellen Sie das erzeugte Layout und die Views auf HAML um
  - Erzeugen Sie ein Layout mit Navigationsleiste auf der linken Seite, ähnlich wie in Aufgabe 03
  - Verwenden Sie eine Definitionsliste „dl, dt, dd“ (!) für eine zweistufige Navigation. Hauptpunkte sind die vier Modelle
    - Autor(inn)en
    - Bücher
    - Buchexemplare
    - Benutzer(innen)
  - Über Unterpunkte wollen wir die jeweiligen „index“-Aktionen verlinken. Dabei sollen **Filter** wirksam werden, die nur bestimmte Objekte zur Anzeige bringen.
    - Autoren: alle / ohne Buch (Autor:in angelegt, aber noch ohne Buch)
    - Bücher: alle / ohne Exemplar (Buch angelegt, aber nicht im Bestand)
    - Buchexemplare: alle / ausleihbar / vorgemerkt / ausgeliehen / überfällig
    - Benutzer: alle / mit Vormerkungen / mit Ausleihen / mit überfälligen Ausleihen

Projekt: Views

- Alle Views
  - Stellen Sie die Texte um auf Deutsch
- Authors
  - Keine besonderen Anforderungen
  - Zuordnung der Bücher erfolgt über „Books“
- Users
  - Keine besonderen Anforderungen
  - Blenden Sie ~~vorläufig~~ das Feld „password“ aus.

Projekt: Views

- Books
  - Hier besteht die Herausforderung im **Berücksichtigen der zugeordneten Autoren/Autorinnen**!
  - index: Spalte „Autoren“ ergänzen, darin Zahl der Autor:innen anzeigen
    - Tipp: `book.authors.count`
  - show: Ausgabe der Autorennamen ergänzen
    (Vorname Nachname, Vorname Nachname, …)
    - Tipp: `@book.authors.map {|a| a.given_name + ' ' + a.family_name}.join(", ")`
  - _form:
    - Auswahl der Autoren über eine Selection-Box mit „multiple“-Attribut
    - Erzeugen bzw. aktualisieren Sie „author_ids“, z.B. mit dem Helper „collection_select“. Bem.: Dieses Vorgehen ist zwar nur bei relativ kleiner Zahl von Autoren sinnvoll, soll hier aber genügen.
    - Hier ist etwas Recherche-Arbeit erforderlich!
    - Hinweis: Wenn Sie „permit“ im Controller verwenden, um eingehende Parameter zu filtern, müssen Sie bei HABTM mitteilen, dass Sie ein Array erwarten, z.B. durch
      Ergänzen von **`{author_ids: []}`**

Projekt: Views

- BookInstances
  - Wir wollen das Verwalten von Stammdaten der Buch-Exemplare von der Verwaltung von Ausleihen und Vormerkungen unterscheiden
  - Dazu führen wir eine neue *action* „lending“ ein und spalten das Formular „_form“ auf in „_form“ und „_form_lending“:

```
$ cp edit.html.haml lending.html.haml

$ cp _form.html.haml _form_lending.html.haml
```

  - Anpassungen im Controller
    - Action „edit“ kopieren und Name der Kopie in „lending“ ändern
    - before_action (Dateianfang): Liste um **`:lending`** erweitern

  - Anpassungen im Routing (config/routes):

```
resources :book_instances do
  member do
    get 'lending'
  end
end
```

Projekt: Views

- BookInstances
  - _form:
    - Auswahl des Buchs über eine Selection-Box
    - Feld „shelfmark“: Signatur normal eingeben (als Textfeld)
    - Feld „purchased_at“: Aktuelles Datum als Voreinstellung
    - Alle anderen Felder entfernen
  - _form_lending:
    - „book“, „shelfmark“: Nur anzeigen, hier nicht änderbar („disabled: true“)
    - „purchased_at“: Hier entfernen
    - Datums- und User-Auswahlen mit geeigneten Helpern. Select-Boxen mit „include_blank: true“
  - index
    - Neue Link-Spalte „Ausleihe“ ergänzen. Tipp:

      ```
      %td= link_to 'Ausleihe', lending_book_instance_path(book_instance)
      ```
  - show: Auch hier ein Link zu „Ausleihe“ ergänzen, analog zu „index“
  - Alle anderen Views:
    - Sinnvoll anpassen, insbesondere Objekt-Referenzen
      - Beispiel: „book.title“ statt nur „book“

Projekt: Controller

- Allgemein
  - Stellen Sie um auf deutsche Mitteilungstexte, die hier ihren Ursprung nehmen
  - Wir ergänzen Filter in den Actions „index“, um jeweils nur eine Auswahl anzuzeigen
  - Definition und Anwahl: Siehe Layout
  - Realisierung: Siehe Model
- Filter
  - Autoren: ohne Buch = Autor angelegt, aber noch ohne Buch
  - Bücher: ohne Exemplar = Buch angelegt, aber nicht im Bestand
  - Buchexemplare:
    - ausleihbar / vorgemerkt / ausgeliehen / überfällig
  - Benutzer:
    - mit Vormerkungen / mit Ausleihen / mit überfälligen Büchern
- Details
  - Technische Hinweise / Details siehe nächste Folien

Projekt: Erweiterungen (Navigation)

- Navigation / Layout: Filter
  - Im Layout
    - Ergänzen Sie Filter-Parameter in den betroffenen Links
    - Beispiel „Autoren“:

```
link_to 'alle', authors_path
link_to 'ohne Buch', authors_path(filter: 'no_books')
```

  - Modulares Vorgehen bei komplizierten Filtern (hier erwünscht):
    - Klassenmethode zur Filterung im Modell anlegen
    - Im Controller nur noch aufrufen (Devise „*Fat models, skinny controllers*“!)
  - Im Controller (Action „index“, Beispiel „Autoren“):

```
@authors = Author.custom_select(params[:filter])
```

  - Im Model (Beispiel „Autoren“):

```
def self.custom_select(filter)       # alternativ mit “scope”
  authors = Author.order('family_name asc')
  case filter
  when 'no_books'
     authors = authors.select {|a| a.books.empty?}
  when …
  end
  return authors
end
```

Projekt: Zwischenstand

- Aktueller Stand
  - Mit den bisherigen Vorgaben können Sie nun
    - einen kleinen Bibliotheksbestand anlegen (4 CRUD-Zyklen, einer pro Modell) sowie
    - Ausleihen und Rückgaben organisieren (über form_lending).
  - Dabei agieren Anwender:innen (noch) in der Rolle der Bibliotheksverwaltung – jede(r) darf noch alles.
- Stammdaten
  - Anregung: Verwenden Sie für Ihre Buch-Stammdaten reale Daten aus dem Lehrbuch-Bestand unserer Fachbibliothek, wenn Sie keine geeigneten Bücher vorliegen haben. Die Daten sind abrufbar, wo Sie auch Fachbücher zur Ausleihe reservieren.
- Konzept
  - Dieser bereits funktionsfähige Zwischenstand ist vergleichsweise leicht erreichbar. Dennoch gibt es dafür bereits vier Punkte für dieses Projekt:
    - Drei Punkte für die elementaren Funktionen incl. korrekte Ausleihe/Rückgabe
    - Ein Punkt für die korrekte Implementierung der Filter
- Ausblick
  - Der folgende Ausblick skizziert Ausbaustufen dieses Projekts, für die uns in diesem Semester die Zeit fehlt, oder die sich für ein weiterführendes Projekt im Rahmen einer Vertiefung anbieten.

Projekt: Ausblick

- Ausblick
  - Models
    - Validations verfeinern
    - Ein paar Spezial-Methoden zur Unterstützung der Views und Filter
    - Mehr Callbacks, z.B. zur Belegung von Default-Werten
    - Authentifizierung, Rollen (user vs. admin), damit verbundene Rechte verwalten
  - Controllers
    - Anmeldung von Benutzern ermöglichen, für Reservierungen (→ Session)
  - Views
    - Tabellen mit jQuery-Bibliothek „datatables“ ausstatten
    - AJAX-Technik für schnelles Reservieren und bequeme Ausleihe/Rückgabe
    - JS/CS-Code für „Accordion“-Verhalten im Navigationsblock (Unterpunkte sind unsichtbar, werden sichtbar jeweils durch Anklicken ihres Haupt-Punktes, Ausblenden aller anderen)
  - Usability
    - Mehrsprachige Seiten, „Mobile first”-Design, allg. Bedienbarbeit steigern, …
  - Tests
    - Systematische, automatisierte Tests ergänzen:
      - Testdaten (fixtures), Model-Tests, Controller-Tests, System-Tests

Projekt: Beispiel

- Beispiel (2 Bildschirm-Ausschnitte)

19.2 ms

Projekt "Fachbibliothek"

- Autor(inn)en
  - alle
  - ohne Buch
- Bücher
  - alle
  - ohne Autoren
  - ohne Buchbestand
- Buchexemplare
  - Gesamtbestand
  - ausleihbare
  - vorgemerkte
  - ausgeliehene
  - überfällige
- Benutzer/innen
  - alle
  - mit Vormerkungen
  - mit Ausleihen
  - mit überfälligen Büchern

Bücherliste

Titel	Verlag	Erscheinungsjahr	# Aut.	# Exemplare	Ausgabe	ISBN			
Rails 5	The Pragmatic Bookshelf	2020	1	1	2	1-234-56789-0	Details	Ändern	Löschen
Rails 6	The Pragmatic Bookshelf	2021	2	0	1	1-234-56791-1	Details	Ändern	Löschen

Neues Buch anlegen

Ausschnitt: Liste der Buchexemplare, mit Zusatzlink zum Ausleihe- und Reservierungsformular

Buchexemplare

Buch	Signatur	Erworben am	Ausgeliehen am	Fällig am	Zurück am	Reserviert				
Rails 5	IT-WBA-1	2021-06-21				Kein User	Ausleihen	Details	Ändern	Löschen

Neues Buchexemplar anlegen

Projekt: Organisatorisches

- Zeitlicher Ablauf
  - Das Projekt erstreckt sich über die restlichen Praktikumstermine. Die Projektergebnisse sind am letzten Praktikumstermin (KW 27) abzugeben und zu präsentieren.
  - Die Bearbeitung erfolgt durch **Zweier-Teams**
- Abgabe:
  - Datei **`Arbeitsaufteilung.txt`** im Basisordner „wba2“ hinzufügen
    - Hier bitte notieren, wer welche Teilarbeiten durchgeführt hat
  - Ausgabe von „rake stats“ in Datei **`stats.txt`** schreiben. In „wba2“:

    ```
    $ rake stats > stats.txt
    ```

  - Putzläufe nicht vergessen:

    ```
    rake webpacker:clobber; rake tmp:clear; rake log:clear
    ```

  - Rails-Projektordner wie immer als komprimiertes tar-Archiv speichern:
  - **07-wba2-`<matnr>`.tar.gz** abgeben

Projekt: Organisatorisches

- Bewertung
  - 4 Punkte (3 für die „basics“ incl. Präsentation, 1 für die Filter)
  - Bis 2 Sonderpunkte für überzeugende und erhebliche Sonderleistungen
    - Beispiel: 1 Sonderpunkt für (relativ vollständige) Tests,
      1 Sonderpunkt für I18n und Mobile first-Design
- Präsentation
  - Max. (!) 10 Min. pro Team! Fokus: Funktionalität, „Highlights“
  - Präsentation per Browser- bzw. Editor-Freigabe im BBB-Raum
    - Falls Tests befüllt: „rake test“ vorführen

Bedingungen

- Abgabe
  - Zum Abnahmetermin, d.h. in der letzten Vorlesungswoche (KW 27)
  - Wie üblich, jeweils vor Beginn Ihrer Praktikumsgruppe
- Art des Leistungsnachweises
  - Zu vergeben: **4 Punkte + eventuelle Sonderpunkte**
    - 3 Punkte für die Grundfunktionen (Daten anlegen&löschen, Ausleihen/Rückgaben mit Sperre)
    - 1 Punkt für den Filter-Teil
    - Zwei mögliche Sonderpunkte für freiwillige Zusatzaufgaben aus dem Ausbau-Teil
  - **Zweier-Teams!**
  - Abgabe der Dateien:
    - Im Verzeichnis „wba2“ ausführen:
      Dateien **`Arbeitsaufteilung.txt`** und **`stats.txt`** erzeugen (s.o.),

      ```
      $ rake webpacker:clobber; rake log:clear; rake tmp:clear
      $ cd ..
      ```
    - Sie sind nun im Elternverzeichnis von „wba2“. Jetzt noch ausführen:

      ```
      # Ordner ./wba2 verpacken, ohne unnötige Unterordner:
      $ tar czf 07-wba2-<matnr>.tar.gz --exclude wba2/tmp \
            --exclude wba2/node_modules --exclude wba2/log  ./wba2
      ```

      Fortsetzungszeile

      Datei „07-wba2-*<matnr>*.tar.gz“ abgeben
    - Achten Sie auf die Entfernung unnötiger Daten – Dateigröße von ca. 400 kB ist ok.
      Dateien >> 1 MB werden nicht angenommen!
  - Online-Präsentation / Online-Demo der korrekten Funktion / Code-Stichproben

Literatur

- **Zu Ruby**
  - **https://ruby-doc.org/core/**
    - Referenz aller eingebauten Klassen und Methoden
  - **https://ruby-doc.org/stdlib/**
    - Referenz der standardmäßig vorhandenen Bibliotheken

- **Zu Rails**
  - **https://guides.rubyonrails.org/**
    - Gute Hilfen für den Einstieg in die Entwicklung mit Rails, mit zahlreichen Beispielen
    - Bereits in 05 empfohlen, immer noch relevant: „Getting started“
    - Sehr hilfreich: Die Guides zu „Model“
    - Optional „Testing Rails Applications“
  - **https://api.rubyonrails.org/**
    - Für alle, die es genau wissen wollen oder müssen

Anhang

Erste Schritte

- Projekt „wba2“ erstellen

```
$ rails new wba2
… (viele Verz.& Dateien)
$ cd wba2
# Gemfile anpassen
# (ggf. haml einbinden,
# Layout umstellen)
# Einfachste Lösung:
$ cp ../wba1/Gemfile .
$ bundle install
```

- 1. Gerüst bauen lassen

```
$ rails g scaffold book
    title:string
    publisher:string
    pub_year:integer
    edition:integer
    isbn:string
… (viele Verz.& Dateien)
```

g = „generate“

Alles in einer Eingabezeile!

- Erster Blick in die DB

```
$ rails dbconsole
.headers ON
.databases
.tables
```

Noch keine Tabellen zu sehen. Beenden mit Strg-D

- Migration-Datei in db/migrate editieren, Vorgaben (z.B. Defaults) umsetzen

- Tabelle generieren (Migration)

```
$ rake db:migrate
```

- DB-Check

```
$ rails dbconsole
.headers ON
.tables
select * from books;
```

Tabelle „books“ nun vorhanden

Tabelle „books“ ist noch leer

Erste Schritte (Forts.)

- Web Server starten

```
$ rails server &
```

- Browser starten
  - http://localhost:3000/
  - http://localhost:3000/books

- CRUD-Zyklus
  - Mehrere Bücher anlegen
    - Ein DB-Check pro Schritt
  - Änderungen, Löschungen
    - Jeweils per DB-Check verfolgen

- Nächstes „Gerüst“ bauen lassen

```
rails g scaffold author
    family_name:string
    given_name:string
    affiliation:text
```

- *Inner join*-Tabelle für die Verknüpfung zwischen Autoren und Büchern:

```
rails g migration authors_books_join_table
```

In db/migrations editieren:

```
class AuthorsBooksJoinTable <
    ActiveRecord::Migration

  def change
    create_join_table :authors, :books do |t|
       t.index :author_id
       t.index :book_id
    end
  end
end
```

- Weitere Tabellen generieren (Migration)

```
$ rake db:migrate
```

- Analoges Vorgehen für die anderen Modelle – Ende der kleinen Starthilfe.