
# Gastvortrag Datenbanksysteme: Ruby on Rails

Nils Haldenwang, B.Sc.

UNIVERSITÄT OSNABRÜCK

Institut für Informatik
AG Medieninformatik

# Models mit ActiveRecord


Ruby on Rails

# Model: ORM

pluralisierter snake_case

Tabelle: users ← Klasse: User

| id | first_name | last_name | age |
|---|---|---|---|
| 1 | Nils | Haldenwang | 25 |
| 2 | Luke | Skywalker | 42 |

snake_case

Idee: Repräsentiere die Tabelle durch eine Klasse und die Zeilen durch einzelne Instanzen.

# Migrations


Ruby on Rails

# Migrations

- Ruby DSL zur Definition und Manipulation des Datenbankschemas
- Datenbankunabhängig (in gewissem Rahmen)
- Inkrementelle Entwicklung der Datenbank
- Kann mit in die Versionskontrolle eingecheckt werden
- Automatische Versionierung
- Gesamtschema in *db/schema.rb*

# Beispiel: Tabelle erstellen

Anpassung des Schemas

Anpassung rückgängig machen

```
class CreateUsers <
    ActiveRecord::Migration

  def up
     create_table :users do |t|
       t.string :first_name
       t.string :last_name

       t.timestamps
     end
  end

  def down
     drop_table :users
  end
end
```

# Model: User

app/models/user.rb

DRY

```ruby
class User < ActiveRecord::Base
end

u = User.new first_name: "Obi-Wan", last_name: "Kenobi"
u.persisted? # => false
u.save
u.persisted? #  => true
puts "#{u.first_name}, #{u.last_name}"
#  => Obi-Wan Kenobi

obi_wan = User.find_by_first_name("Obi-Wan")
puts obi_wan.last_name #  => Kenobi
```

# Query-API


Ruby on Rails

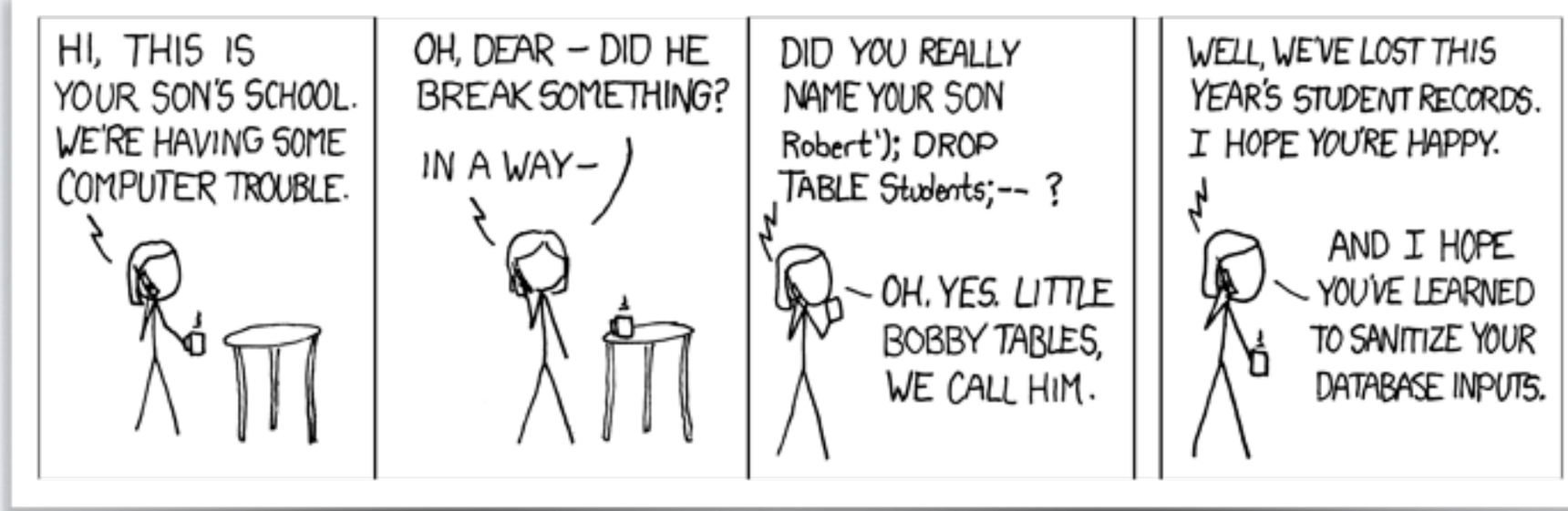
# Where-Clauses

```
User.create name: "Nils", age: 24
User.create name: "Luke", age: 42

User.where(name: "Nils")
User.where("name = ?", "Nils")
User.where("name = :name", name: "Nils")

User.where(name: "Nils", age: 24)
User.where(name: "Nils").where(age: 24)
User.where("name = ? AND age = ?", "Nils", 24)

User.where("name = ? OR age = ?", "Nils", 42)

User.where(name: ["Nils", "Luke"]).
 order("name ASC")

User.where("name LIKE 'L%'")
```

# SQL-Injections

Obacht!

```ruby
User.where("name = #{params[:name]}")
```




```
Robert'); DROP TABLE Students; --
```

# Associations




Ruby on Rails

# 1:1-Beziehung: Definition

Fremdschlüssel: *user_id*

```
class Avatar < ActiveRecord::Base
  belongs_to :user
end

class User < ActiveRecord::Base
  has_one :avatar
end
```

# 1:1-Beziehung: Anwendung

```
u = User.create name: 'Nils'
u.avatar # => nil
a = Avatar.new url: 'foo.jpeg'
a.persisted? # => false
u.avatar = a
a.persisted? #  => true

u.avatar = Avatar.new url: 'bar.jpeg'

Avatar.all
# => [
#       #<Avatar id: 1, url: "foo.jpeg", user_id: nil>,
#       #<Avatar id: 2, url: "bar.jpeg", user_id: 1>
#    ]
```

# Abhängigkeiten

```
class Avatar < ActiveRecord::Base
  belongs_to :user
end

class User < ActiveRecord::Base
  has_one :avatar, dependent: :destroy
end
```

# 1:n-Beziehung: Definition

Fremdschlüssel: *user_id*

```ruby
class Article < ActiveRecord::Base
  belongs_to :user
end

class User < ActiveRecord::Base
  has_many :articles
end
```

# 1:n-Beziehung: Anwendung

```ruby
u = User.first
u.articles #  => nil
u.articles.new title: "Title"
# => #<Article id: nil, title: "Title", user_id: 1>
u.save
u.articles.first
#  => => #<Article id: 1, title: "Title", user_id: 1>

u.articles << Article.new( title: "Another Title" )
u.articles.count #  => 2

u.articles.delete Article.find(1)
u.articles.count #  => 1

u.articles.clear
u.articles.count #  => 0

Article.count #  => 2
```

# n:m-Beziehung: Definition

```ruby
class Article < ActiveRecord::Base
  has_and_belongs_to_many :tags
end

class Tag < ActiveRecord::Base
  has_and_belongs_to_many :articles
end
```

```ruby
class CreateArticlesTags < ActiveRecord::Migration
  def change
    create_table :articles_tags, id: false do |t|
      t.references :article, null: false
      t.references :tag, null: false
    end
  end
```

Sortiert

# Validations

# Validations: Grundlagen

```ruby
class User < ActiveRecord::Base
  validates_numericality_of :age
  validates_presence_of :name
end

u = User.new age: "foo"
u.valid? #  => false
u.errors[:age] # => "is not a number"
u.errors[:name] # => "can't be blank"
```

# Validations: Parameter

```
validates_presence_of :name, message: "Y u no use name?"

validates_numericality_of :age, on: :create

validates_length_of :name, within: 6..42

validates :name, presence: true,
                 length: { minimum: 6 },
                 uniqueness: true
```

# Validierung mit Methode

```
class Feed < ActiveRecord::Base
  validate :valid_feed_url

  protected
  def valid_feed_url
    result = Feedzirra::Feed.fetch_and_parse(self.url)
    if result.nil? || result.kind_of?(Fixnum)
      errors.add(:url, "The feed url was not valid.")
    end
  end
end
```

# Callbacks

# Callback-Registrierung

```ruby
class Beethoven < ActiveRecord::Base
  before_destroy :last_words

  protected
  def last_words
    logger.info "Friends applaud, the comedy is over."
  end
end

class Beethoven < ActiveRecord::Base
  before_destroy do
    logger.info "Friends applaud, the comedy is over."
  end

  before_save(on: create) do
    logger.info "Here we go."
  end
end
```

# Mögliche Callbacks

- before_validation, before_validation_on_create
- after_validation, after_validation_on_create
- before_save
- before_create, after_create
- before_destroy, after_destroy

# Beispiel: Geocoding

```
class Address < ActiveRecord::Base
  include GeoKit::Geocoders

  before_save :geolocate
  validates_presence_of :street, :city, :state, :zip

  def to_s
    "#{street} #{city} #{state} #{zip}"
  end

  protected
  def geolocate
    res = GoogleGeocoder.geocode(to_s)
    self.lattitude = res.lat
    self.longitude = res.lng
  end
end
```

# Verarbeitung abbrechen

```
def geolocate
  res = GoogleGeocoder.geocode(to_s)
  if res.success
    self.lattitude = res.lat
    self.longitude = res.lng
  else
    errors[:base] = "Geocoding failed. Please check address."
    false
  end
end
```

Abbruch der Verarbeitung, wenn
ein Callback *false* liefert

# Views mit HTML & ERB

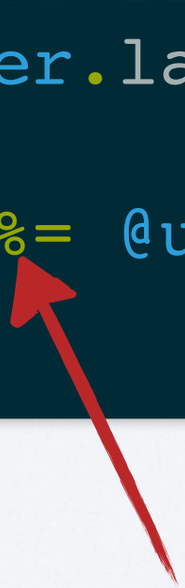


# HTML-Gerüst

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Titel der Webseite</title>
    <!-- Kommentare werden nicht angezeigt. -->
  </head>
  <body>
    <div id="header" class="container">
      <h1>Überschrift</h1>
      <p>Inhalt <em>der</em> Webseite</p>
    </div>
  </body>
</html>
```

# Embedded Ruby (ERB)

```
<div id='user-<%= @user.id %>'>
  <h2>
    <%= @user.first_name %>
    <%= @user.last_name %>
  </h2>
  <p>Age: <%= @user.age %></p>
</div>
```



Einbetten des Rückgabewertes
eines Ruby-Ausdruckes

# ERB und Schleifen

```
<div id="squares">
  <% 1.upto(3) do |i| %>
    <p> The square of <%= i %> is <%= i*i %>.</p>
  <% end %>
</div>
```

Ausführung ohne Einbettung

```
<div id="squares">
    <p> The square of 1 is 1.</p>
    <p> The square of 2 is 4.</p>
    <p> The square of 3 is 9.</p>
</div>
```

# Controller & Routing


Ruby on Rails

# Controller: UsersController

app/controllers/users_controller.rb

```ruby
class UsersController < ApplicationController
  def show
    @user = User.find(params[:id])
  end
end
```

app/config/routes.rb

```ruby
MyRailsApp::Application.routes.draw do
  get 'users/:id' => "users#show"
end
```

# RESTful Controller Actions

| HTTP Verb | URI | Action |
|---|---|---|
| GET | /users | index |
| POST | /users | create |
| GET | /users/:id | show |
| PUT | /users/:id | update |
| DELETE | /users/:id | destroy |
| GET | /users/new | new |
| GET | /users/:id/edit | edit |

# Demo: Blog


Ruby on Rails

# Anforderungen: Blog

- Ein Blog besteht aus Artikeln
- Artikel gehören zu einer Kategorie
- Ein Artikel kann mit mehreren Schlagworten versehen sein