Agile Webentwicklung mit Ruby on Rails

Prof. Dr. Oliver Vornberger
Nils Haldenwang, B.Sc.

UNIVERSITÄT OSNABRÜCK

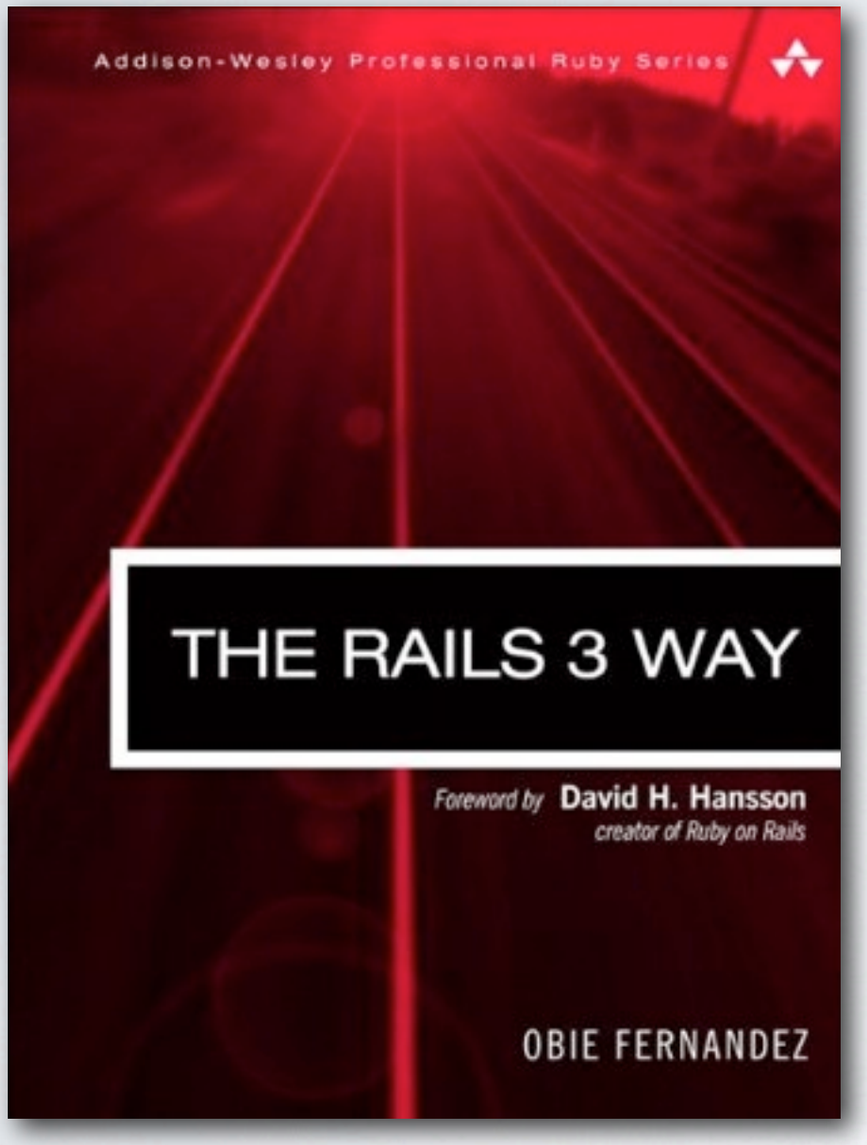
Institut für Informatik
AG Medieninformatik

Ruby on Rails

Konfiguration

Ruby on Rails

Quellen/Literatur



Obie Fernandez,
The Rails 3 Way,
Addison-Wesley Professional,
2nd Edition, 2010

S. 1-29.

Konfiguration

- Verwaltung von Abhängigkeiten mit Bundler
- Vorkonfigurierte Umgebungen
  - config/environments/development.rb
  - config/environments/test.rb
  - config/environments/production.rb
- Initializer zur Initialisierung bestimmter Komponenten

Dependency Management mit Bundler

http://gembundler.com/

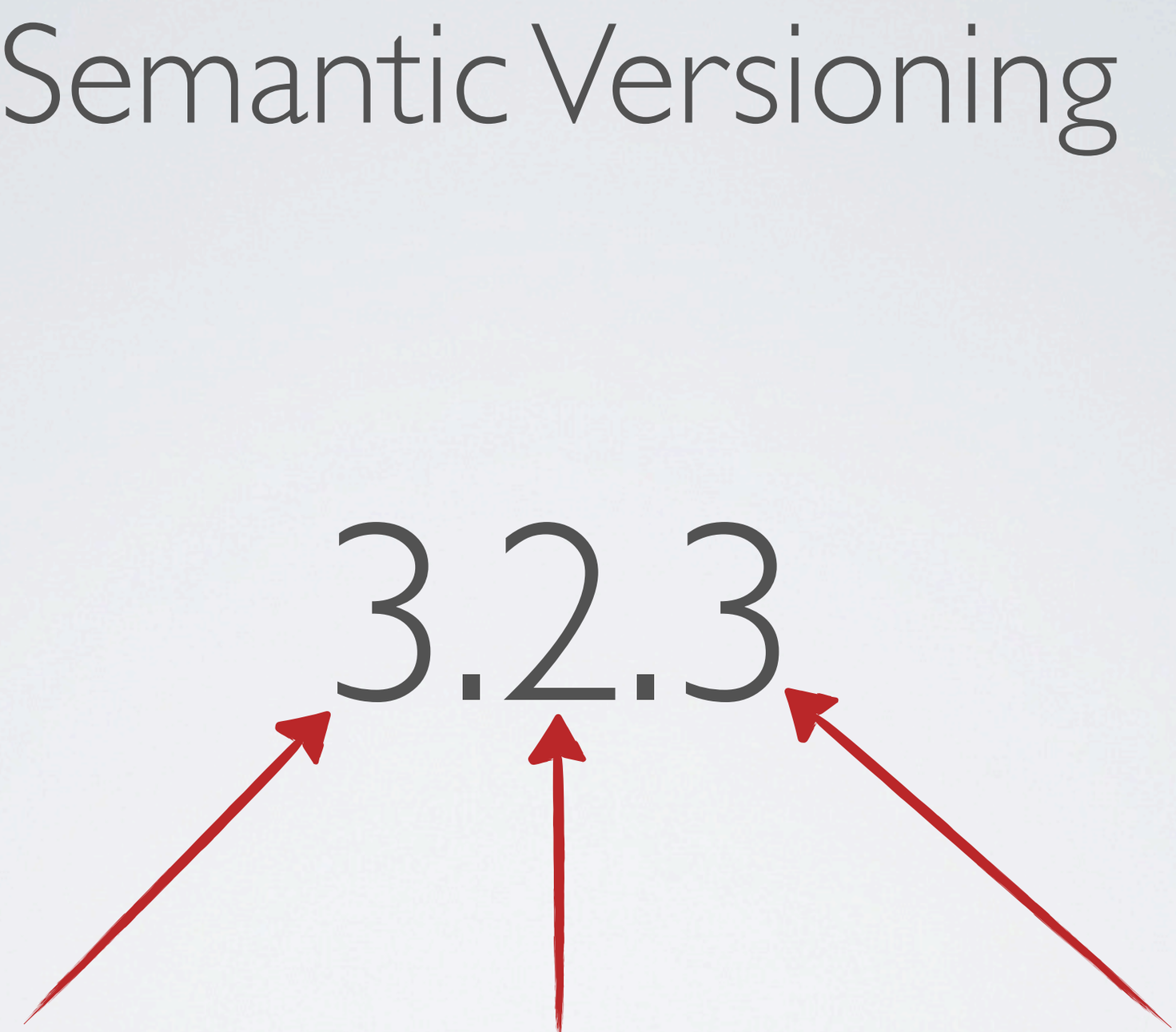
Problem: Gems hängen von verschiedenen Versionen anderer Gems ab

Idee: Prüfe die Abhängigkeiten aller Gems gleichzeitig

```
$ gem install bundler
```

Semantic Versioning

3.2.3



Bruch mit der Rückwärtskompatibilität

Neue Features

Bug-Fixes

Gemfile

```
source "http://rubygems.org"

gem "nokogiri"

gem "rack", ">= 1.1"

gem "faker", "~> 0.3"

gem "paperclip",
  git: "git://github.com/
thoughtbot/paperclip.git"

gem "rspec", require: "spec"
```

Aktuellste, stabile Version

Minimale Version

Nur letzte Zahl aktualisieren

Git als Quelle

Alternatives Require

Gemfile: Gruppen

```
source "http://rubygems.org"

group :production do
  gem "new_relic"
end

group :development do
  gem "pry-rails"
end

group :test do
  gem "rspec"
  gem "faker"
end

group :development, :test do
  gem "wirble”
end
```

Gemfile.lock

```
$ bundle install
```

```
GEM
  remote: http://rubygems.org/
  specs:
    activemodel (3.2.5)
      activesupport (= 3.2.5)
      builder (~> 3.0.0)
    activerecord (3.2.5)
      activemodel (= 3.2.5)
      activesupport (= 3.2.5)
      arel (~> 3.0.2)
      tzinfo (~> 0.3.29)
    activesupport (3.2.5)
```

Gem Dependencies für nicht-Rails Skripte

```
$ bundle exec cucumber
```

Startup

- config/boot.rb
  - Bindet Bundler ein und konfiguriert den LOAD_PATH
- config/application.rb
  - Lädt Gems und konfiguriert die Applikation
- config/environment.rb
  - Führt alle Initializer aus

Auto-Loading

```
# config/application.rb
config.autoload_paths += [ "#{config.root}/extras" ]
```

```
KittTurboBoost => require 'kitt_turbo_boost'

MagGyver::SwissArmyKnife
 => require 'mac_gyver/swiss_army_knife'

Example::ReallyRatherDeeply::NestedClass
 => require 'example/really_rather_deeply/nested_class'
```

Initializer: Inflections

```
ActiveSupport::Inflector.pluralize "project" # => projects
"project".pluralize #  => projects
"virus".pluralize # => viri
```

```
# config/initializers/inflections.rb
ActiveSupport::Inflector.inflections do |inflect|
  inflect.plural /^(ox)$/i, '\1en'
  inflect.singular /^(ox)en$/i, '\1'
  inflect.irregular 'person', 'people'
  inflect.uncountable %w( fish sheep )
end
```

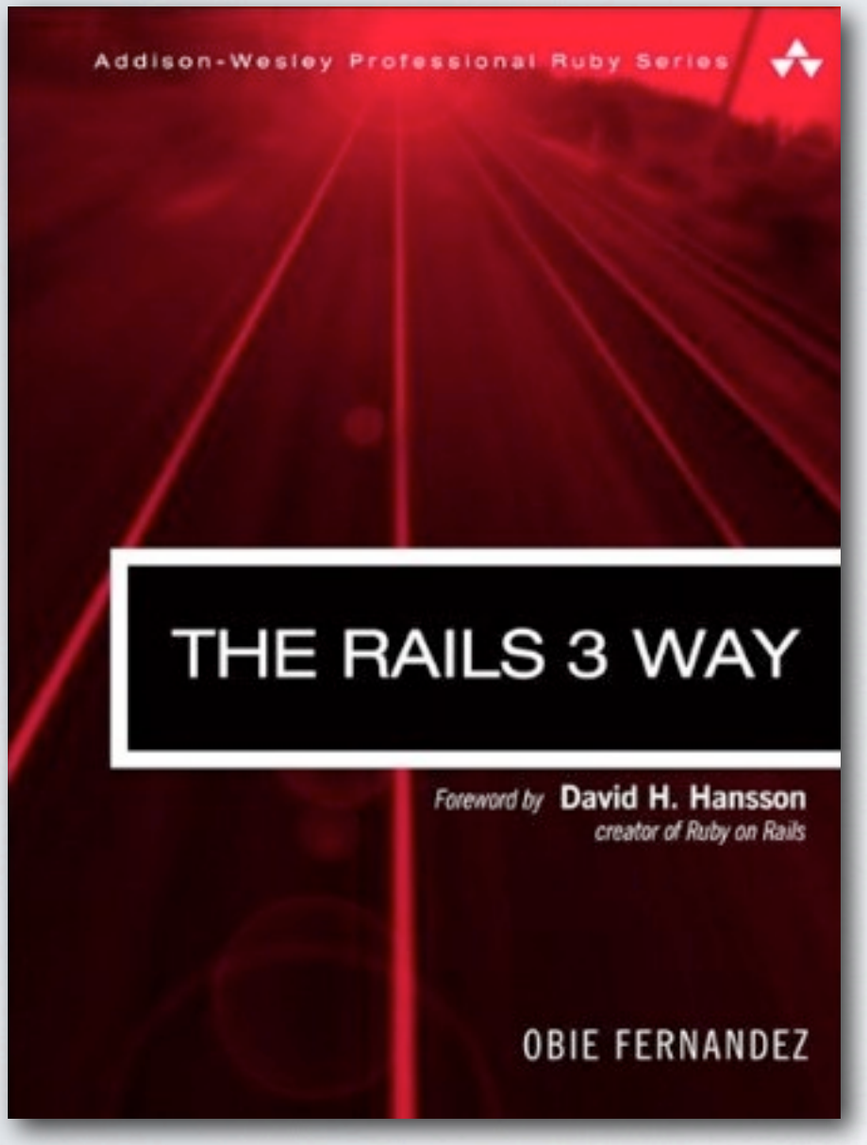
ActiveRecord

Grundlagen

Ruby on Rails

Associations

Quellen/Literatur



Obie Fernandez,
The Rails 3 Way,
Addison-Wesley Professional,
2nd Edition, 2010

S. 181-230.

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

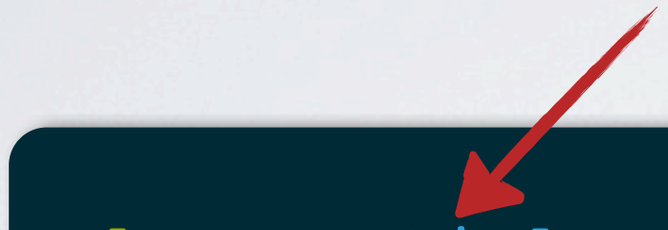
```
u = User.new name: 'Nils'
u.avatar # => nil
a = Avatar.new url: 'foo.jpeg'
a.persisted? # => false
u.avatar = a
a.persisted? #  => true

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
end
```

Sortiert

n+1-Problem

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

class Article < ActiveRecord::Base
  belongs_to :user
  belongs_to :category
end

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

n+1-Problem

```
<h1><%= @user.name %></h1>

<h2>Articles</h2>
<ul>
  <% @user.articles.each do |a| %>
    <li>
      <%= link_to "#{a.title} (#{a.category.name})", a %>
    </li>
  <% end %>
</ul>
```

Bei n Artikeln sind n zusätzliche Queries nötig,
um die Kategorien zu laden

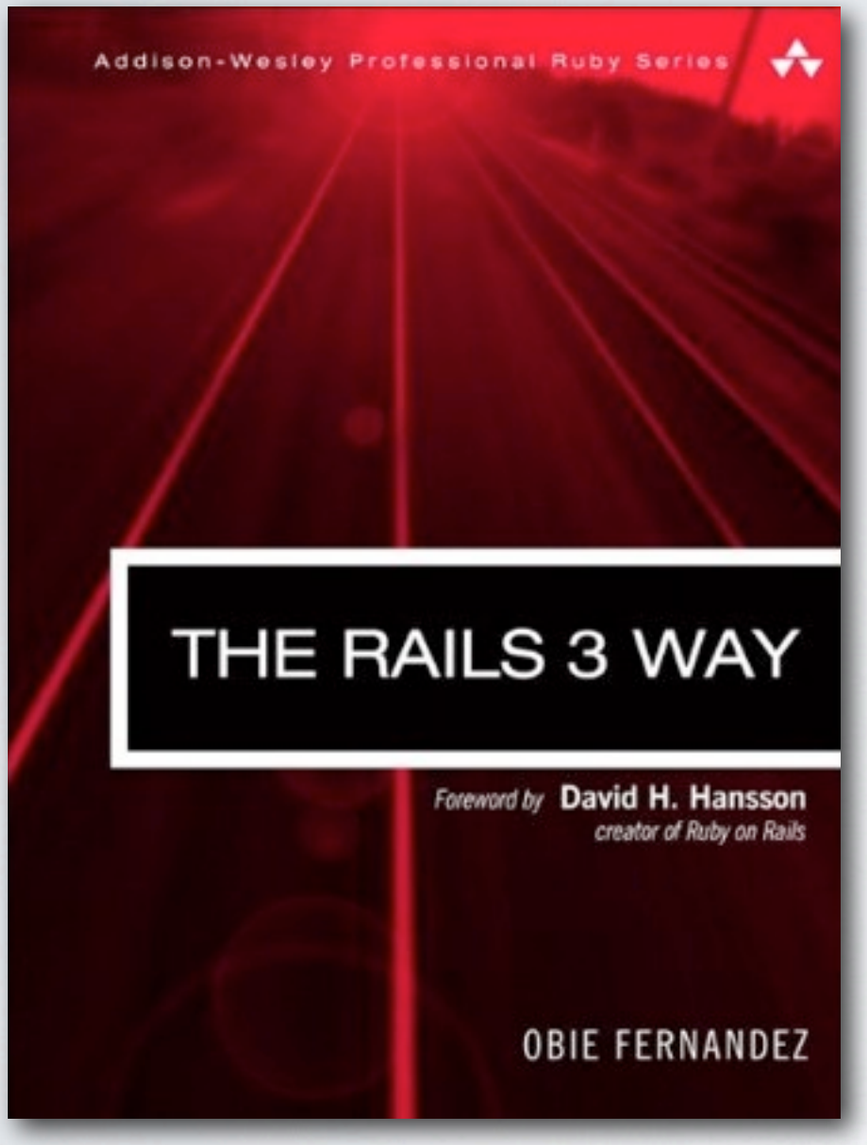
n+1-Problem: Lösung

```ruby
class UsersController < ActionController::Base
  def show
    @user = User.find(params[:id])
    @articles = @user.articles.includes(:category)
  end
end
```

```erb
<h1><%= @user.name %></h1>

<h2>Articles</h2>
<ul>
  <% @articles.each do |a| %>
    <li>
      <%= link_to "#{a.title} (#{a.category.name})", a %>
    </li>
  <% end %>
</ul>
```

Query-API

Quellen/Literatur



Obie Fernandez,
The Rails 3 Way,
Addison-Wesley Professional,
2nd Edition, 2010

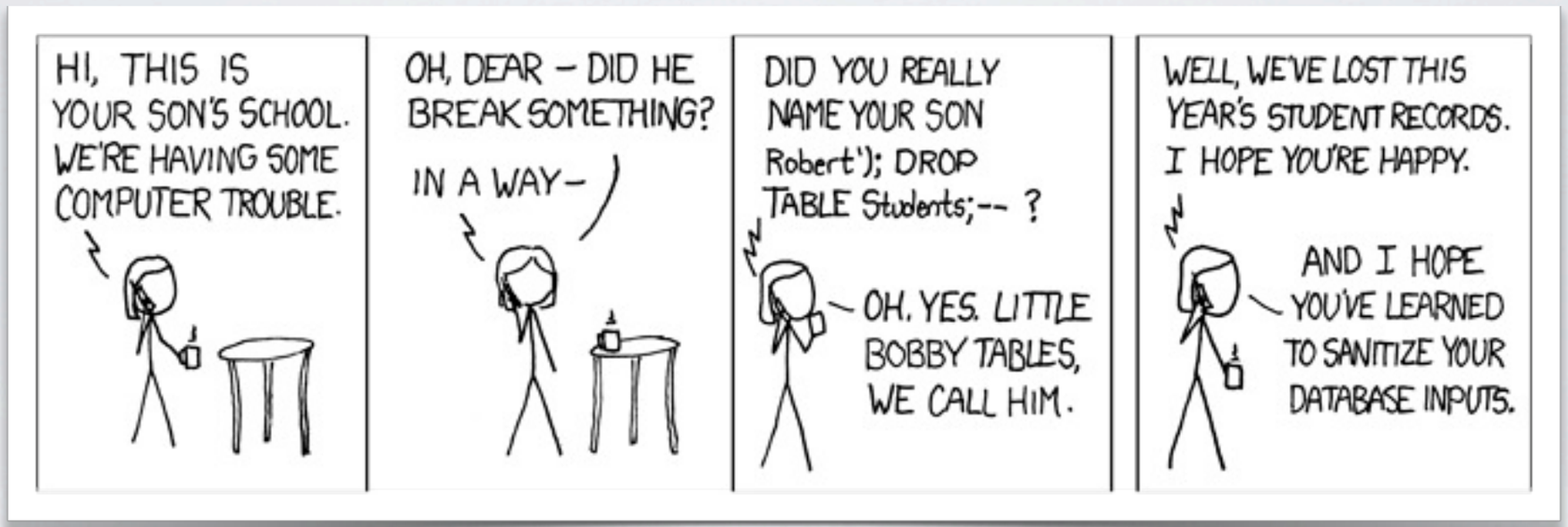
S. 146-152.

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"])

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

SQL-Injections

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




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

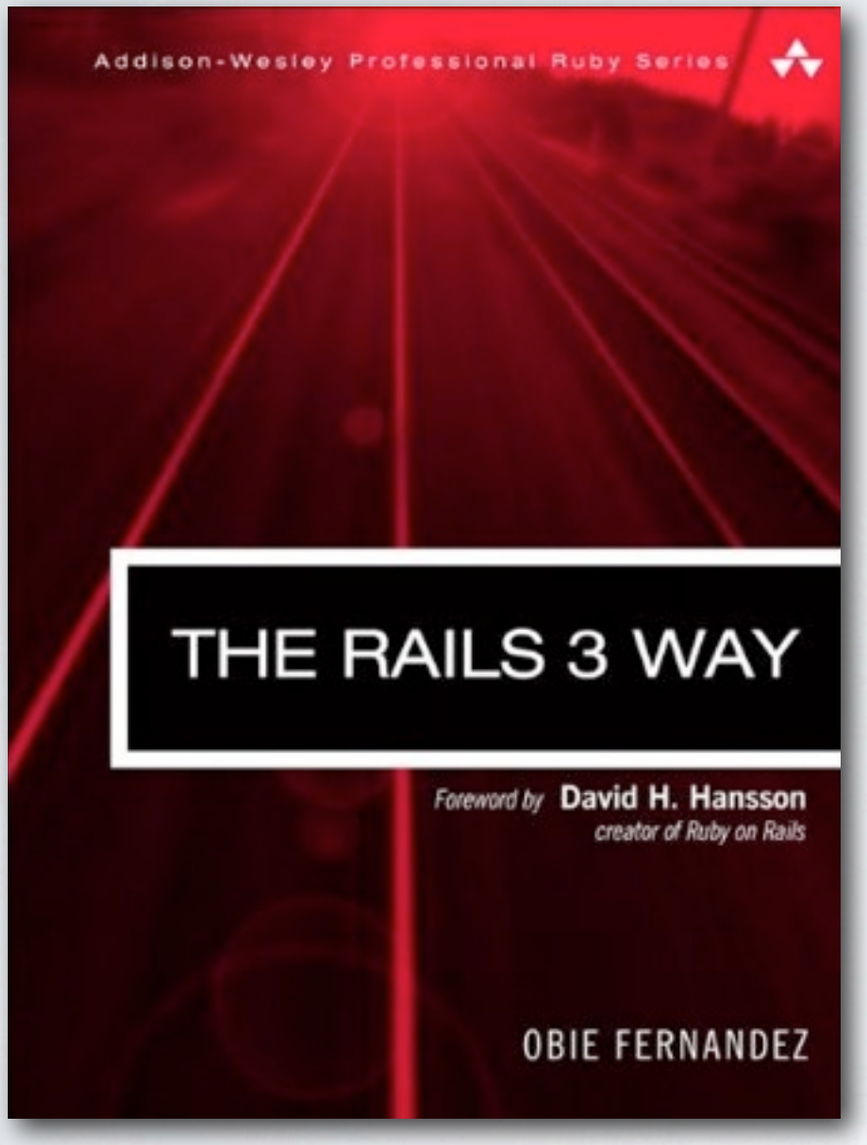
Sortierung

```
User.order("name ASC")

User.order("name DESC")
```

Obacht: Keine Default-Sortierung!

Validations

Quellen/Literatur



Obie Fernandez,
The Rails 3 Way,
Addison-Wesley Professional,
2nd Edition, 2010

S. 231-250.

Validations: Grundlagen

```ruby
class User < ActiveRecord::Base
  attr_accessible :age, :name
  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

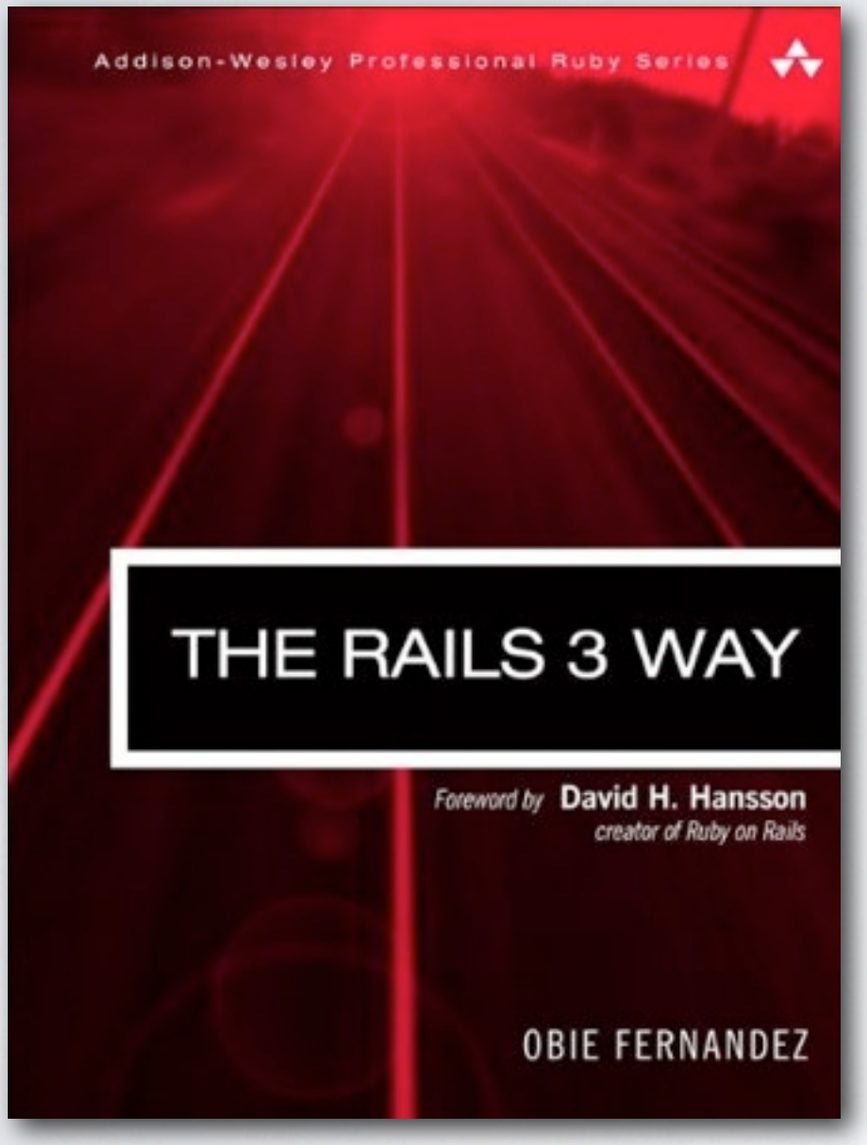
```
validates_presence_of :name, message: "Y u no use name?"

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

Quellen/Literatur



Obie Fernandez,
The Rails 3 Way,
Addison-Wesley Professional,
2nd Edition, 2010

S. 256-264.

Callback-Registrierung

```
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

Demo