

The background is a collage of various icons related to web development and technology. It includes a hand cursor pointing at a screen, a globe, a target, a magnifying glass, a notepad, a smartphone, a heart, a keyboard, a document with a checkmark, and a document with a greater-than sign. The icons are in shades of gray and are arranged in a circular pattern around the central text.

# Web-Entwicklung 2

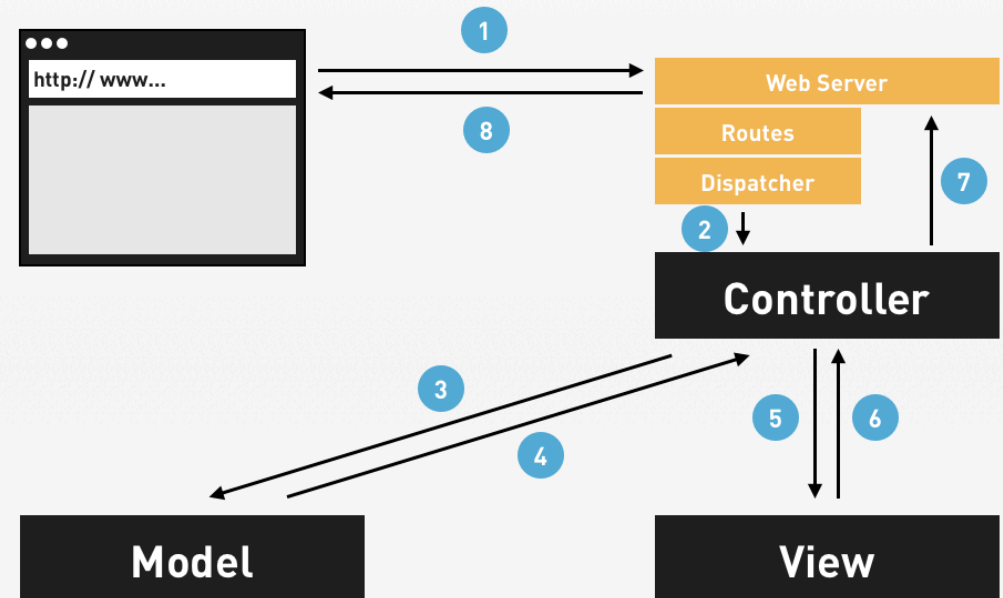
## Vorlesung 3

Fachbereich Wirtschaft - Fachhochschule Münster  
Bachelor of Science Wirtschaftsinformatik Wintersemester 2013/2014

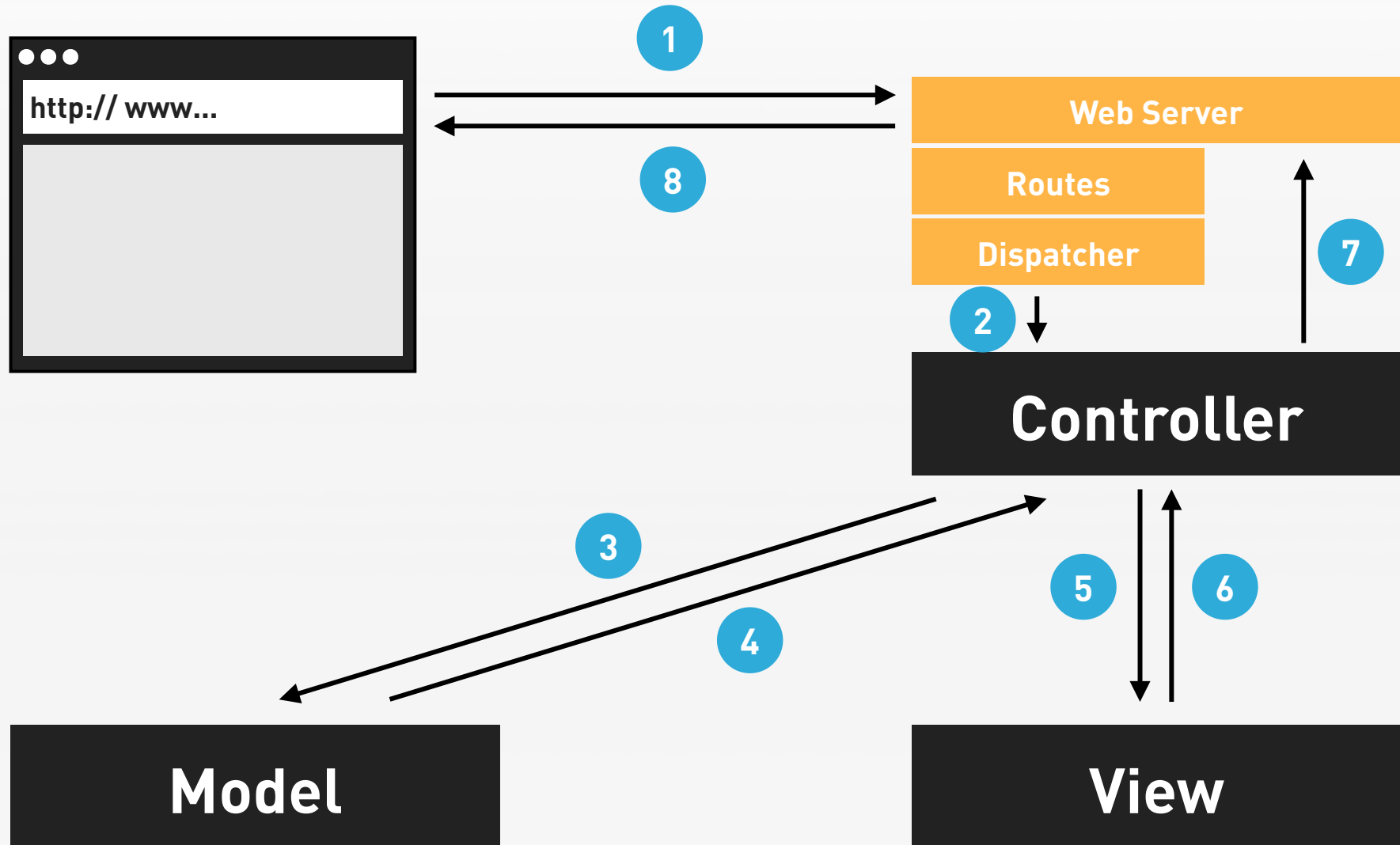
# Was war?

- Rails ist ein Entwicklungsframework für Webanwendungen basierend auf Ruby
- Rails hat Prinzipien
- Rails ermöglicht schnellen Einstieg durch Scaffolding
- Rails implementiert das MVC-Entwurfsmuster

## MVC - Realisierung in Rails



# MVC - Realisierung in Rails



# Erzeugen einer Rails-Anwendung

`$ rails new blog`

Erzeugt eine neue Rails-Anwendung mit Namen **blog** und installiert benötigte Pakete

## Scaffolding

**Scaffolding** (deut. Gerüstbau) ist eine Technik der Meta-Programmierung, bei der automatisch ein **Quellcode-Grundgerüst** mit gewissen Basisfunktionalitäten generiert wird. Dieses kann und soll dann vom Entwickler angepasst und erweitert werden.



# Rails-Prinzipien

**"Don't repeat yourself" (DRY) -**

Design-Prinzip für Software-Architekturen, das besagt, dass Informationen möglichst nicht redundant an mehreren Stellen im Quellcode vorgehalten werden sollen.

"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." (Hunt & Thomas, The Pragmatic Programmer, 1999)

# Rails-Prinzipien

## "Convention over Configuration" (CoC) -

Design-Prinzip für Software-Komponenten, das darauf abzielt, durch gut gewähltes Standardverhalten die Arbeit des Entwicklers im "Normalfall" zu minimieren, ohne jedoch die für besondere Fälle notwendige Flexibilität einzubüßen.

Rails trifft vernünftige Annahmen und benötigt keine aufwändige Konfiguration





# Aufbau von Rails

# Aufbau von Rails

- Action Pack
  - Action Controller
  - Action Dispatch
  - Action View
- Action Mailer
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

siehe: [http://guides.rubyonrails.org/getting\\_started.html](http://guides.rubyonrails.org/getting_started.html)





# **Erzeugung einer Ressource**

# Scaffold: Erzeugen einer Ressource

- Rails enthält einen Ressourcen-Scaffold
- Erzeugt sämtlichen, für eine Ressource nötigen Quellcode
  - Model
  - Controller
  - Views
  - Helper
  - ...
- Erzeugter Quellcode muss i.d.R. angepasst werden

# Scaffold: Erzeugen einer Ressource

- Syntax:

```
$ rails generate scaffold Name attribute1:type  
attribute2:type [...]
```

- Beispiel:

```
$ rails generate scaffold Post name:string  
title:string content:text
```

# Scaffold: Erzeugen einer Ressource

```
thomas@t420: ~/blog
thomas@t420 ~/blog (git)-[master] % rails generate scaffold Post name:string title:string content:text
active_record
  create db/migrate/20121024164036_create_posts.rb
  create app/models/post.rb
  test_unit
    create test/unit/post_test.rb
    create test/fixtures/posts.yml
resource_route
  route resources :posts
scaffold_controller
  create app/controllers/posts_controller.rb
  erb
    create app/views/posts
    create app/views/posts/index.html.erb
    create app/views/posts/edit.html.erb
    create app/views/posts/show.html.erb
    create app/views/posts/new.html.erb
    create app/views/posts/_form.html.erb
  test_unit
    create test/functional/posts_controller_test.rb
  helper
    create app/helpers/posts_helper.rb
    test_unit
      create test/unit/helpers/posts_helper_test.rb
assets
  coffee
    create app/assets/javascripts/posts.js.coffee
  scss
    create app/assets/stylesheets/posts.css.scss
  scss
    create app/assets/stylesheets/scaffolds.css.scss
thomas@t420 ~/blog (git)-[master] %
```

# Scaffold: Migration

```
db/migrate/20121024164036_create_posts.rb

class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :name
      t.string :title
      t.text :content
      t.timestamps
    end
  end
end
```

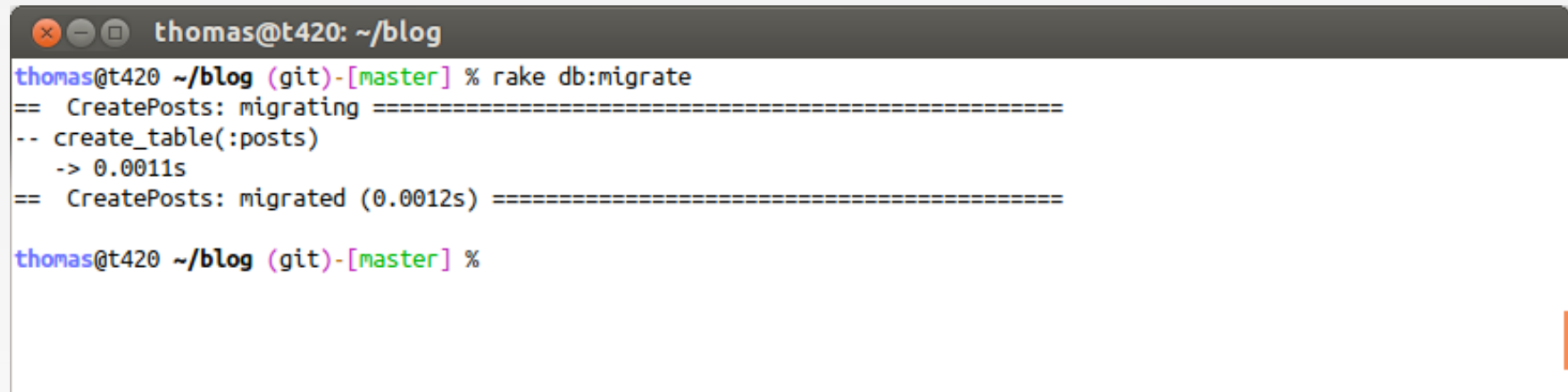
## Migration

Migrations sind Ruby-Klassen, die Anpassungen an einem Datenbank-Schema vornehmen. Sie können einzeln ausgeführt und zurückgenommen werden ("Rollback"). Design-Prinzip für Software-Komponenten, das darauf



# Scaffold: Migration

- Migrationen werden durch einen rake-Task ausgeführt
- Beispiel: `$ rake db:migrate`
- Rails merkt sich, welche Migrations bereits ausgeführt wurden

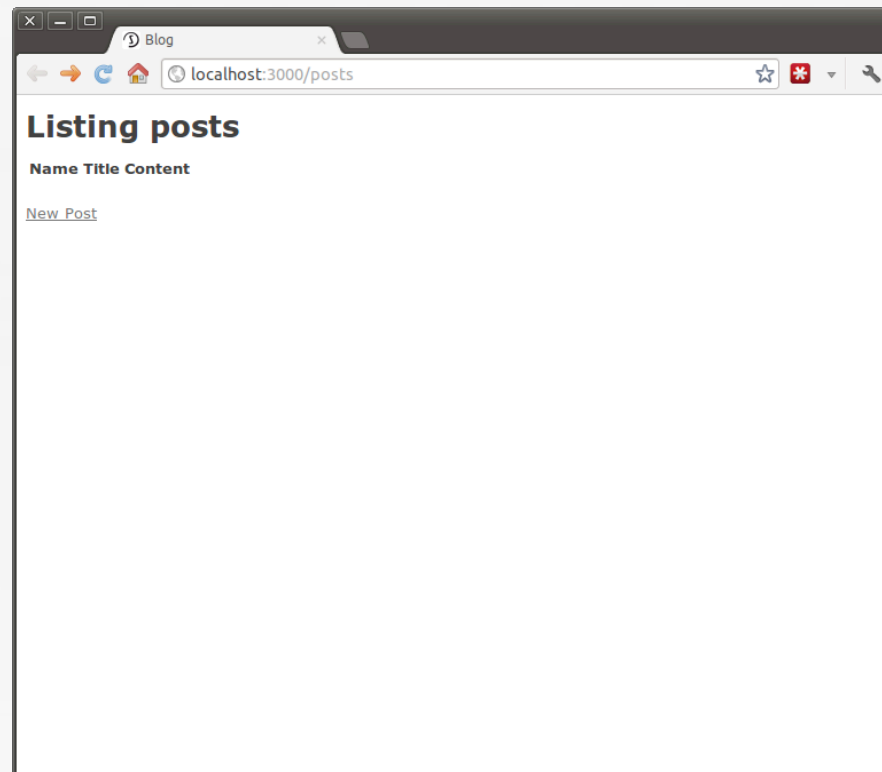


```
thomas@t420: ~/blog
thomas@t420 ~/blog (git)-[master] % rake db:migrate
== CreatePosts: migrating =====
-- create_table(:posts)
   -> 0.0011s
== CreatePosts: migrated (0.0012s) =====

thomas@t420 ~/blog (git)-[master] %
```

# Scaffold: Das Ergebnis

- Starten des Rails-Servers: `$ rails server`
- Öffnen der URL <http://localhost:3000/posts>



# Scaffold: Das Ergebnis



The screenshot shows a web browser window with the address bar displaying 'localhost:3000/posts/new'. The page title is 'New post'. The form contains three input fields: 'Name' with the value 'Hello Rails', 'Title' with the value 'Dies ist ein erster Beitrag', and 'Content' with the value 'Willkommen auf meinem neuen Blog! Hier veröffentliche ich interessante Artikel zum Thema Ruby on Rails.' Below the form is a 'Create Post' button and a 'Back' link.

**New post**

Name

Title

Content

[Back](#)

# Scaffold: Das Ergebnis



- Einfache Anwendung zur Verwaltung von Posts
- Kein Design
- Keine komplexe Funktionalität



# **Scaffold - Outside In**



# Scaffold: Routen

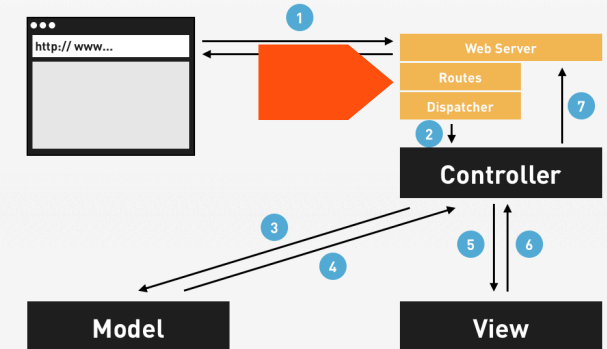
```
config/routes.rb

Blog::Application.routes.draw do
  # ...

  resources :posts
end
```

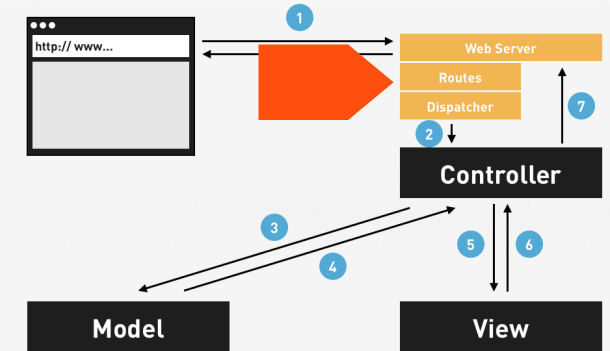
- Syntax: resources :<Model (Plural)>
- Erzeugt Routen für CRUD-Operationen
  - CRUD: Create, Read, Update, Delete
- Erzeugt Path-Helper für Routen
  - später mehr dazu!

## MVC - Realisierung in Rails



# Scaffold: Routen

## MVC - Realisierung in Rails

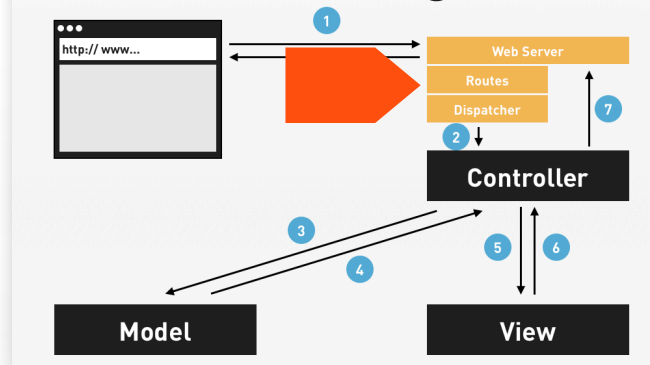


HTTP Verb	Pfad	Action	Beschreibung
GET	/posts	index	Zeigt eine Liste aller Posts an
POST	/posts	create	Erstellt einen neuen Post
GET	/posts/new	new	Zeigt ein HTML-Formular zur Erstellung eines Posts an
GET	/posts/:id/edit	edit	Zeigt ein HTML-Formular zur Bearbeitung eines Posts an
GET	/posts/:id	show	Zeigt einen einzelnen Post an
PUT	/posts/:id	update	Ändert einen einzelnen Post
DELETE	/posts/:id	destroy	Löscht einen einzelnen Post

# Scaffold: Routen

- Rails bietet einen Rake-Task zur Anzeige aller erzeugten Routen an
- Syntax: `rake routes`

## MVC - Realisierung in Rails



```
thomas@t420: ~/blog
thomas@t420 ~/blog (git)-[master] % rake routes
welcome_index GET    /welcome/index(.:format) welcome#index
root           /                  welcome#index
posts GET        /posts(.:format)      posts#index
               POST       /posts(.:format)      posts#create
new_post GET     /posts/new(.:format)  posts#new
edit_post GET    /posts/:id/edit(.:format) posts#edit
post GET       /posts/:id(.:format)  posts#show
               PUT       /posts/:id(.:format)  posts#update
               DELETE  /posts/:id(.:format)  posts#destroy
thomas@t420 ~/blog (git)-[master] %
```

# Scaffold: Controller

```
app/controllers/posts_controller.rb

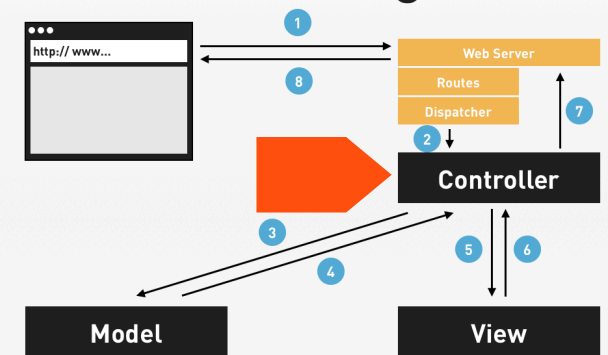
class PostsController < ApplicationController

  def index
    # ...
  end

  def show
    @post = Post.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.json { render json: @post }
    end
  end

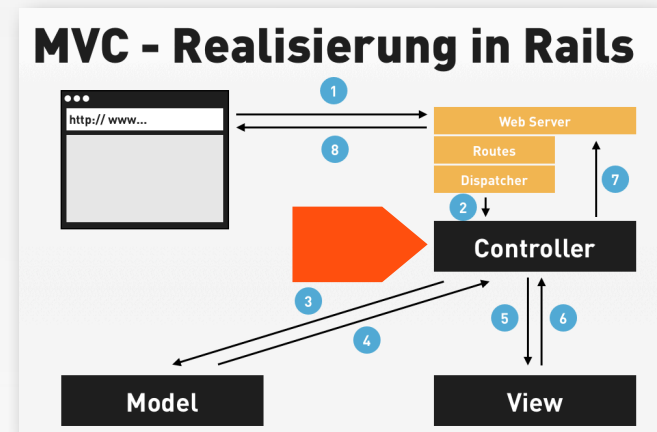
  # ...
end
```

## MVC - Realisierung in Rails



# Scaffold: Controller

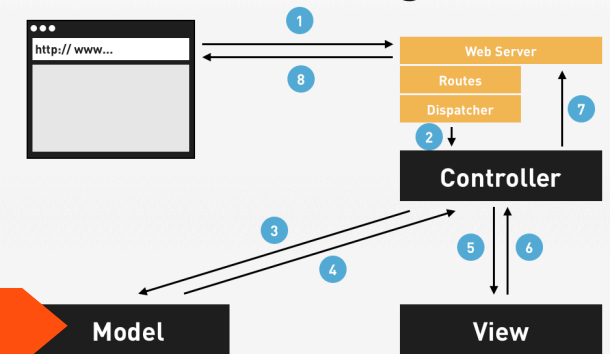
- Controller enthält CRUD-Actions
  - Actions: Instanzmethoden der Controller-Klasse
- Actions enthalten Interaktionen mit Models und Views
  - Model-Informationen werden aus der Datenbank geladen
  - Informationen werden an die View weitergereicht





# Scaffold: Model

## MVC - Realisierung in Rails



```
app/models/post.rb

class Post < ActiveRecord::Base
  attr_accessible :content, :name, :title
end
```

- Model kapselt Zugriff auf Datenbank
- Stellt Hilfsmethoden zum Finden von Posts bereit
- Erzeugt automatisch Getter und Setter für Attribute
  - mehr dazu später!

# Scaffold: Views

app/views/posts/show.html.erb

```
<p id="notice"><%= notice %></p>
```

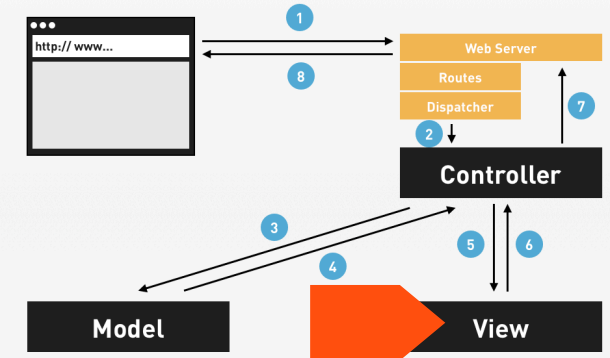
```
<p>  
  <b>Name:</b> <%= @post.name %>  
</p>
```

```
<p>  
  <b>Title:</b> <%= @post.title %>  
</p>
```

```
<p>  
  <b>Content:</b> <%= @post.content %>  
</p>
```

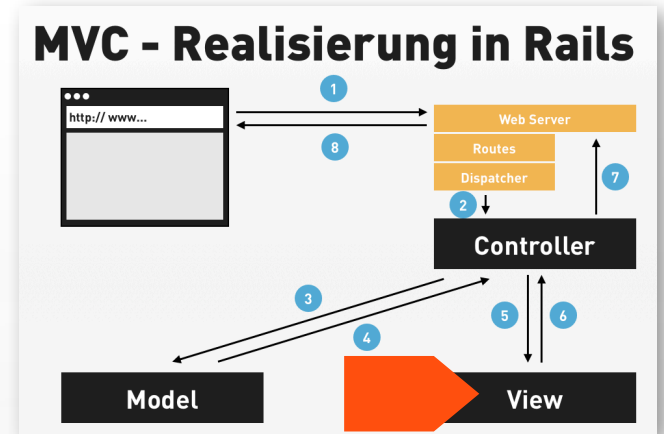
```
<%= link_to 'Edit', edit_post_path(@post) %> |  
<%= link_to 'Back', posts_path %>
```

## MVC - Realisierung in Rails



# Scaffold: Views

- Erzeuger Scaffold enthält eine View pro Controller-Action
- View zeigt vom Controller bereitgestellte Informationen an
  - Beispiel: `<%= @post.name %>`
- View greift auf Helper zu
  - Beispiel: `<%= link_to 'Back', posts_path %>`
  - Path-Helper werden automatisch aus Routen erzeugt



# Scaffold: Weitere Dateien

- Javascript:

- `app/assets/javascripts/posts.js.coffee`
- Hier kann Post-spezifisches Javascript hinterlegt werden

- Stylesheets:

- `app/assets/stylesheets/posts.css.scss`
- `app/assets/stylesheets/scaffolds.css.scss`
- Stylesheet-Dateien für Post-Seiten und Scaffolds

# Scaffold: Weitere Dateien

- **Helper:**

- `app/helpers/posts_helper.rb`

- Hier können eigene Hilfsmethoden für Views abgelegt werden

- **Test-Dateien**

- `test/`

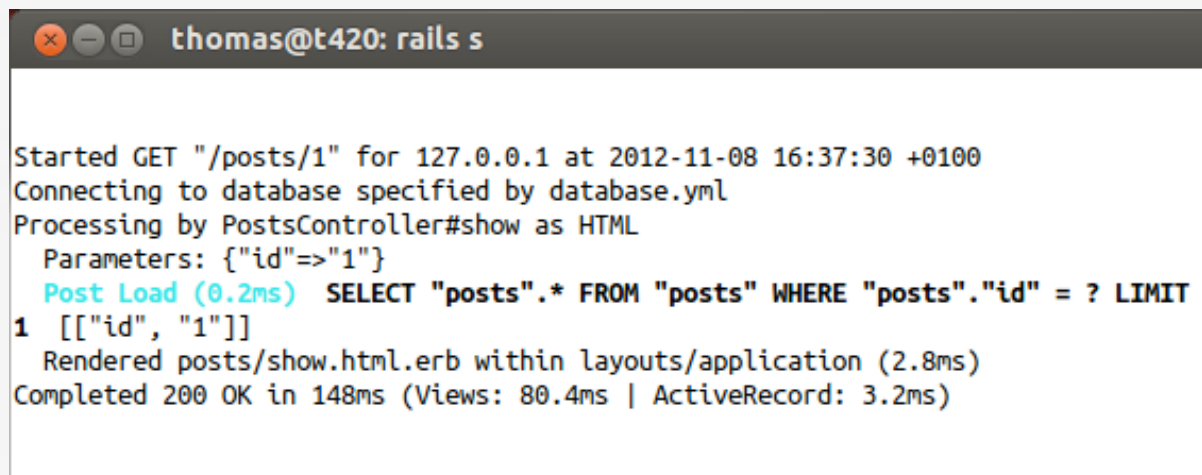
- Test-Dateien für Models, Views und Controller

- Werden für Test-Driven Development (TDD) genutzt



# Rails Scaffold in Action

- Ausgabe des Rails-Servers enthält Informationen über alle durchlaufenen Schritte eines Requests
- Beispiel: Aufruf von <http://localhost:3000/posts/1>



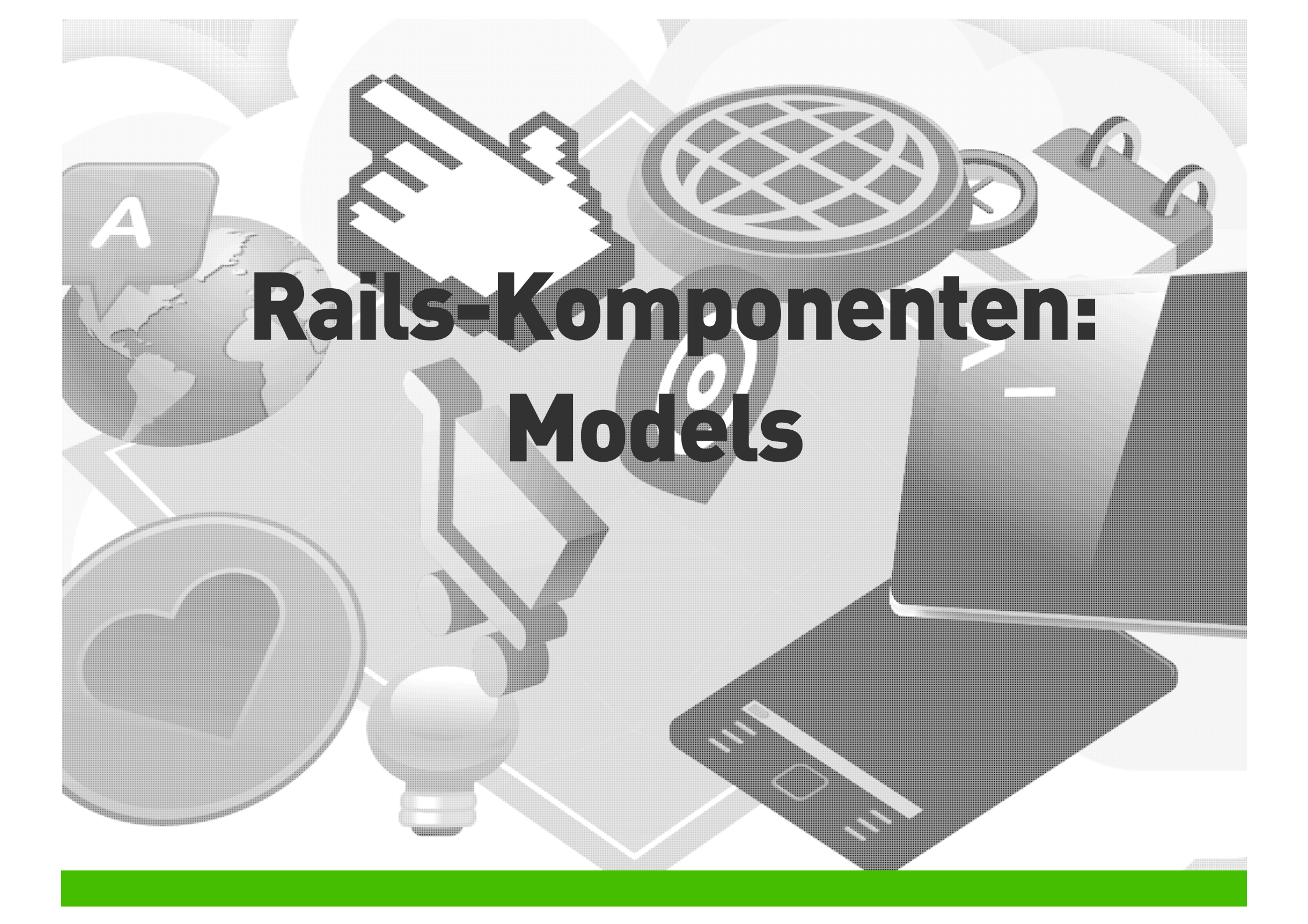
```
thomas@t420: rails s

Started GET "/posts/1" for 127.0.0.1 at 2012-11-08 16:37:30 +0100
Connecting to database specified by database.yml
Processing by PostsController#show as HTML
Parameters: {"id"=>"1"}
Post Load (0.2ms) SELECT "posts".* FROM "posts" WHERE "posts"."id" = ? LIMIT
1  [["id", "1"]]
Rendered posts/show.html.erb within layouts/application (2.8ms)
Completed 200 OK in 148ms (Views: 80.4ms | ActiveRecord: 3.2ms)
```

# Rails Scaffold in Action

```
Started GET "/posts/1" for 127.0.0.1 at 2012-11-08
16:40:22 +0100
Processing by PostsController#show as HTML
  Parameters: {"id"=>"1"}
  Post Load (0.1ms)  SELECT "posts".* FROM "posts"
  WHERE "posts"."id" = ? LIMIT 1  [["id", "1"]]
  Rendered posts/show.html.erb within layouts/
  application (1.8ms)
Completed 200 OK in 10ms (Views: 9.0ms |
ActiveRecord: 0.1ms)
```

**Webserver**  
**Routing**  
**Controller**  
**Model (SQL)**  
**View**  
**Webserver**



# **Rails-Komponenten: Models**

# Models: ActiveRecord

- ORM (Object-Relational Mapper)

- Übersetzt Operationen auf Objekten in Datenbankabfragen

- Grundfunktionen

- Suchen, speichern und löschen in der Datenbank
- Erzeugung von Gettern und Settern

- Erweiterte Funktionen

- Assoziationen
- Validierungen
- Scopes

...

→ Mehr dazu im Verlauf der Veranstaltung!

# Exkurs: Arbeiten auf der Konsole

- Rails liefert eine Eingabeaufforderung mit, in der Rails-Befehle ausgeführt werden können
- Start: `$ rails console`
- Beispiel:



```
thomas@t420: rails console
thomas@t420 ~/blog (git)-[master] % rails console
Loading development environment (Rails 3.2.8)
1.9.3p125 :001 > Post.first
Post Load (0.1ms) SELECT "posts".* FROM "posts" LIMIT 1
=> #<Post id: 1, name: "Hello Rails", title: "Dies ist ein erster Beitrag", content: "Willkommen auf meinem neuen Blog! Hier veröffentlic...", created_at: "2012-10-24 17:00:56", updated_at: "2012-10-24 17:00:56">
1.9.3p125 :002 > █
```

# Models: Grundfunktionen

Suchen eines Objekts anhand der ID:

> `Post.find(1)`

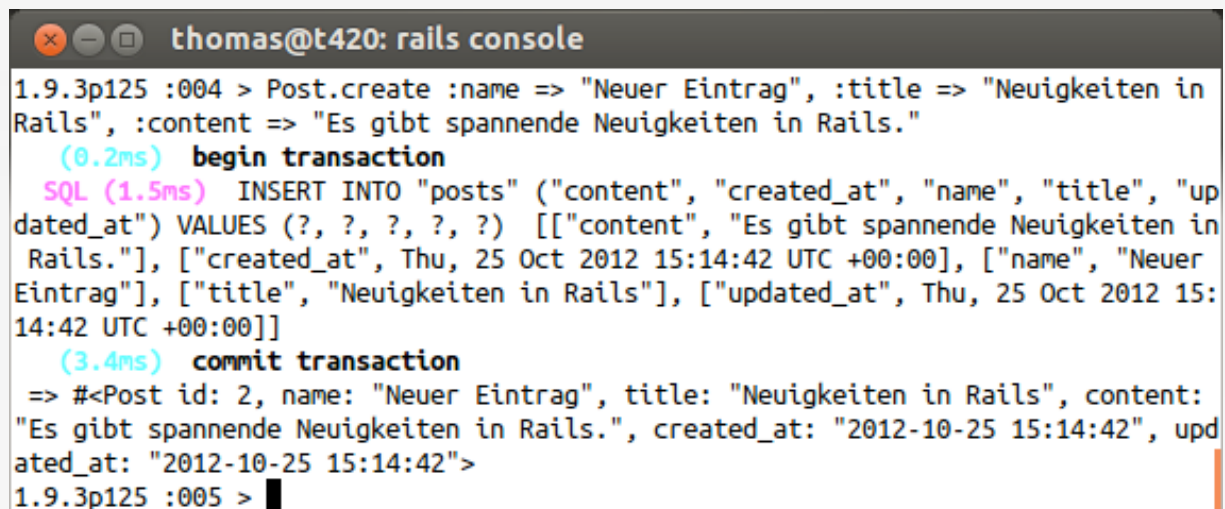
A screenshot of a terminal window titled "thomas@t420: rails console". The prompt is "1.9.3p125 :002 >". The user enters "Post.find(1)". The output shows a SQL query: "Post Load (9.3ms) SELECT 'posts'.\* FROM 'posts' WHERE 'posts'.'id' = ? LIMIT 1" followed by "[["id", 1]]". Then it shows the object: "=> #<Post id: 1, name: 'Hello Rails', title: 'Dies ist ein erster Beitrag', content: 'Willkommen auf meinem neuen Blog! Hier veröffentlic...', created\_at: '2012-10-24 17:00:56', updated\_at: '2012-10-24 17:00:56'>". The prompt changes to "1.9.3p125 :003 >".

```
thomas@t420: rails console
1.9.3p125 :002 > Post.find(1)
Post Load (9.3ms) SELECT "posts".* FROM "posts" WHERE "posts"."id" = ? LIMIT 1
[["id", 1]]
=> #<Post id: 1, name: "Hello Rails", title: "Dies ist ein erster Beitrag", content: "Willkommen auf meinem neuen Blog! Hier veröffentlic...", created_at: "2012-10-24 17:00:56", updated_at: "2012-10-24 17:00:56">
1.9.3p125 :003 >
```

# Models: Grundfunktionen

Erstellen eines Objekts:

> `Post.create :name => "Neuer Eintrag", :title => "Neuigkeiten in Rails", :content => "Spannende News"`



```
thomas@t420: rails console
1.9.3p125 :004 > Post.create :name => "Neuer Eintrag", :title => "Neuigkeiten in Rails", :content => "Es gibt spannende Neuigkeiten in Rails."
(0.2ms) begin transaction
SQL (1.5ms) INSERT INTO "posts" ("content", "created_at", "name", "title", "updated_at") VALUES (?, ?, ?, ?, ?) [["content", "Es gibt spannende Neuigkeiten in Rails."], ["created_at", Thu, 25 Oct 2012 15:14:42 UTC +00:00], ["name", "Neuer Eintrag"], ["title", "Neuigkeiten in Rails"], ["updated_at", Thu, 25 Oct 2012 15:14:42 UTC +00:00]]
(3.4ms) commit transaction
=> #<Post id: 2, name: "Neuer Eintrag", title: "Neuigkeiten in Rails", content: "Es gibt spannende Neuigkeiten in Rails.", created_at: "2012-10-25 15:14:42", updated_at: "2012-10-25 15:14:42">
1.9.3p125 :005 > █
```



# Models: Grundfunktionen

Löschen eines Objekts:

> `post = Post.find(2)`

> `post.destroy`



```
thomas@t420: rails console
1.9.3p125 :002 > post.destroy
(0.2ms) begin transaction
SQL (0.7ms) DELETE FROM "posts" WHERE "posts"."id" = ? [["id", 2]]
(3.5ms) commit transaction
=> #<Post id: 2, name: "Neuer Eintrag", title: "Neuigkeiten in Rails", content:
"Es gibt spannende Neuigkeiten in Rails.", created_at: "2012-10-25 15:19:20", upd
ated_at: "2012-10-25 15:19:20">
1.9.3p125 :003 > █
```

# Models: Grundfunktionen

Erzeugen von Gettern und Settern

```
> post = Post.find(1)
```

```
> post.name
```

```
# => "Hello Rails"
```

```
> post.name = "Hallo Welt"
```

```
# => "Hallo Welt"
```

```
> post.save
```

```
# => true
```