Tag 1, Hands-On 1: Getting started with Ruby

Ziel des Hands-on	Ruby Syntax verstehen und in den Grundzügen schreiben können.
Aufgabe	1. Gib auf vier verschiedene Arten den String “Hello World!” *viermal* auf dem Bildschirm aus. Die unterschiedlichen Methoden sollten möglichst kreativ sein! 2. Eine Liste von Studenten, die Matrikelnummern den Namen der Studenten zuweist (mind. 4 Studenten in der Liste). Aus dieser Liste sollen die Namen sortiert ausgegeben werden.
Ressourcen	www.ruby-doc.org/core Ruby Cheat Sheet

Shortcuts	**Strings:** `“string”, ‘string’`
	Strings: `:key`
	Arrays: `[1,2,3], %w(worte stehen hier)`
	Hash: `{‘key’ => ‘value’, ‘another_key’ => ‘just_a_value’}`
	Ranges: `0..4 => [0,1,2,3]`
	Schleifen: `4.times { |n| puts n }`
	`while i < 10`
	`  # do something`
	`end`
	`[1,2,3].each { |x| puts x+1 }`
	`for i in 0..10 do |n|`
	`  # do something with ‘n’`
	`end`
	Ausgabe: `puts string`
	Ruby Source Dateien: `my_ruby_prog.rb`
	Ruby Interpreter starten: `ruby <Dateiname>`
	Interactive Ruby Shell starten: `irb`
	Rückgabewerte: Eine Funktion gibt immer den zuletzt evaluierten Wert zurück, auch ohne Angabe des Schlüsselworts `` `return` ``.
	Konventionen:
	• Klassen-/Modulnamen werden in `CamelCase` notiert
	• Methodenamen werden mit Unterstrichen notiert (`find_by_name`)
	• Methoden, die das Callerobjekt verändern haben einen Ausrufezeichen (`sort!`)
	• Methoden, die true oder false zurückgeben haben ein Fragezeichen (`instance_of?`)

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Ruby Cheatsheet

v.1 for Ruby 1.8.4

Types

12345
123.45
1.23e-4
0xFF00
0b01100
1..5
1...5
‚a'..'z'
‚a'...'z'
‚*string sq*'
„*string dq*“
„#{*expr*}“
„\t\r\n“
%q(*string sq*)
%Q(*string dq*)
%(*string dq*)
<<id *string* id
:*symbol*
/*regex*/*opt*
%r|*regex*|
[1, 2, 3]
%w(1 2 3)
%W(1 2 #{*expr*})
{1=>2, :s=>'v'}

Exceptions

begin
rescue *ex* =>
var
else
ensure
end

StandardError
ZeroDivisi- onError
RangeError
SecurityError
IOError
IndexError
RuntimeError

Expressions

if *expr* [then] elsif *expr* [then] else end
unless *expr* [then] else end
expr if *expr*
expr unless *expr*
case *expr* when comp else end
while *expr* [do] end
until *expr* [do] end
do while *expr*
do until *expr*
for *var* in *expr* [do] end
expr.each [do] end
break next redo retry

Module/Class

module *Name* end
class *Name* end
class *Name* < *Sup* end
class << *obj* end
def *name*(*args*...) end
def *inst.* *name*(...) end
public protected private
attr_reader attr_writer attr attr_accessor
alias *new old*

Variables

local
@instance
@@class
CONSTANT

Operators and Precedence

::
[]
**
+ - ! ~
* / %
<< >>
&
| ^
> >= < <=
<=> == === != =~
&&
.. ...
= (+= -=)
not
and or

Constants

__FILE__
__LINE__
ENV
ARGF
ARGV

Predefined Variables

$!	Exception information
$@	Array of backtrace
$&	String of last match
$`	String left of last match
$'	Str right of last match
$+	Last group of last match
$N	Nth group of last match
$~	Info about last match
$=	Case insensitive flag
$/	Input record separator
$\	Output record separator
$,	Output field separator
$.	Line number of last file
$>	Default output
$_	Last input line of string
$*	Command line args
$0	Name of script
$$	Process number
$“	Module names loaded
$stderr	Standard error output
$stdin	Standard input
$stdout	Standard output

Regex

.	all characters
[ ]	any single char in set
[^]	any single char not in set
*	zero or more
+	one or more
?	zero or one
|	alteration
( )	Group
^	Beginning of line or str
$	End of line or string
{1,5}	1 to 5
\A	Beginning of a string
\b	Word boundary
\B	Non-word boundary
\d	digit, same as [0..9]
\D	Non-digit
\s	Whitespace
\S	Non-whitespace
\w	Word-character
\W	Non-word-character
\z	End of a string
\Z	End of string, before nl

Ruby arguments

-c	Check
-d	Debug
-e	One Line
-h	Help
-n	gets loop
-rL	require L
-v	verbose
-w	warnings
-y	comp debug

Reserved Words

alias
and
BEGIN
begin
break
case
class
def
defined?
do
else
elsif
END
end
ensure
false
for
if
in
module
next
nil
not
or
redo
rescue
retry
return
self
super
then
true
undef
unless
until
when
while
yield

Object

Obj#class -> class
Obj#freeze -> object
Obj#frozen? -> true or false
Obj#inspect -> string
Obj#is_a? (*class*) -> true or false
Obj#methods -> array
Obj#respond_to? (*sym*) -> true or false
Obj#to_s -> string

String

Str#[*num*, *num/range/regx*] -> str
Str#capitalize! -> string
Str#center (*int* [,*str*]) -> str
Str#chomp! ([*str*]) -> str
Str#count -> integer
Str#delete! ([*string*]) -> string
Str#downcase! -> string
Str#each ([*str*]) do |str| ... end
Str#each_line do |line| ... end
Str#gsub! (*rgx*) do |match| ... end
Str#include? (*str*) -> true / false
Str#index (*str/reg* [,*off*]) -> int
Str#insert (*int*, *string*) -> string
Str#length -> integer
Str#ljust (*int* [,*padstr*]) -> str
Str#rindex (*str/reg* [,*off*]) -> int
Str#rjust (*int* [,*padstr*]) -> str
Str#scan (*rgx*) do |match| ... end
Str#split (*string*) -> array
Str#strip! -> string
Str#sub! (*rgx*) do |match| ... end
Str#swapcase! -> string
Str#to_sym -> symbol
Str#tr! (*string*, *string*) -> string
Str#upcase! -> string

Kernel

block_given?
eval (*str [,binding]*)
raise (*exception [,string]*)
fork do ... end => fixnum or nil
proc do ... end => proc
print (*obj*)
warn (*msg*)

Array

Array::new (*int* [,*obj*]) -> array
Array#clear
Array#map! do |x| ... end
Array#delete (*value*) -> obj or nil
Array#delete_at (*index*)-> obj or n
Array#delete_if do |x| ... end
Array#each do |x| ... end
Array#flatten! -> array
Array#include? (*value*) -> t or f
Array#insert (*idx*, *obj*...)-> array
Array#join ([*string*]) -> string
Array#length -> integer
Array#pop -> obj or nil
Array#push (*obj*...) -> array

Hash

Hash#clear
Hash#delete (*key*) -> obj or nil
Hash#delete_if do |*k*, *v*| ... end
Hash#each do |*k*, *v*| ... end
Hash#has_key? (*k*) -> true or false
Hash#has_value? (*v*) -> t or f
Hash#index (*value*) -> key
Hash#keys -> array
Hash#length -> integer
Hash#select do |*k*, *v*| ... end -> array
Hash#values -> array

Test::Unit

assert (*boolean [,msg]*)
assert_block (*message*) do ... end
assert_equal (*expected, actual [,msg]*)
assert_in_delta (*exp, act, dlt [,message]*)
assert_kind_of (*klass, object [,msg]*)
assert_match (*pattern, string [,msg]*)
assert_nil (*object [,msg]*)
assert_no_match (*pattern, string [,msg]*)
assert_not_equal (*expected, actual [,msg]*)
assert_not_nil (*object [,msg]*)
assert_not_same (*expected, actual [,msg]*)
assert_respond_to(*obj, method [,msg]*)
assert_same (*expected, actual [,msg]*)

File

File#new (*path*, *modestring*)-> file
File#new (*path*, *modestring*) do |file| ... end
File#open (*path*, *modestring*) do |file| ... end
File#exist? (*path*) -> t or f
File#basename (*path* [,*suffix*]) -> string
File#delete (*path*, ...)
File#rename (*old*, *new*)
File#size (*path*) -> integer

r	Read-only, from beginning
r+	Read-write, from beginning
w	Write-only, trunc. / new
w+	Read-write, trunc. / new
a	Write-only, from end / new
a+	Read-write, from end / new
b	Binary (Windows only)

Dir

Dir[*string*] -> array
Dir::chdir ([*string*])
Dir::delete (*string*)
Dir::entries (*string*) -> array
Dir::foreach (*string*) do |file| ... end
Dir::getwd -> string
Dir::mkdir (*string*)
Dir::new (*string*)
Dir::open (*string*) do |dir| .. end
Dir#close
Dir#pos -> integer
Dir#read -> string or nil
Dir#rewind

DateTime

DateTime::now
DateTime::parse (str)
DateTime::strptime (str, format)
DateTime#day
DateTime#hour
DateTime#leap?
DateTime#min
DateTime#month
DateTime#sec
DateTime#wday
DateTime#year

Ruby: www.ruby-lang.org
Doc: www.ruby-doc.org

Tag 1, Hands-On 2: Getting started with Rails

Ziel des Hands-on	Ein Rails Projekt aufsetzen können und die Datenbank einrichten
Aufgabe	1. Es soll die Projektstruktur für die "MyBlog" Applikation angelegt werden. Anschließend soll als erstes die Datenbank angelegt werden, anschließend soll das erste Modell der Applikation erzeugt werden. Als Hilfestellung dient das angefügte Flowchart der Blog-Applikation. Am Ende sollen die CRUD Funktionalitäten für das erzeugte Modell verfügbar sein und über den Browser ansprechbar sein. Mögliche Felder für einen Blogeintrag wären zum Beispiel: - Titel - Author (Name / Email) - Inhalt 2. Damit beim Aufruf der URL direkt die Liste aller Blogeinträge angezeigt wird, und nicht die „Welcome aboard" Nachricht von Rails, soll eine Default Route eingerichtet werden.
Ressourcen	http://www.ruby-doc.org/core http://api.rubyonrails.com Ruby on Rails Cheat Sheet Rails Architektur Datenbank Informationen: • Server (für alle): mysql.local • username: ass_user<Nummer> | password: O2U4ASS • Datenbankname: ass<Nummer>_myblog_(dev|test|prod)
Shortcuts	**Rails Projektstruktur initialisieren:** `rails <appname>` **Datenbank Config:** `RAILS_ROOT/config/database.yml` **Scaffold Generator:** `RAILS_ROOT/script/generate scaffold <Model>` **Server starten:** `RAILS_ROOT/script/server start` **Konventionen:** • Tabellen sind immer in Pluralform • Primary Key ist `id` und vom Typ `int(n)` • ActiveRecord Klassen sind immer im Singular • ActionController Klassen sind immer im Plural **Wichtige Dateien:** • Templates: `RAILS_ROOT/app/views/<ControllerName>/<action>.rhtml` • Controller: `RAILS_ROOT/app/controllers/<ControllerName>_controller.rb` • Modelle: `RAILS_ROOT/app/models/<ModelName>.rb` • Konfiguration: `RAILS_ROOT/config/` • Routing Konfiguration: `RAILS_ROOT/config/routes.rb` • Statische Inhalte: `RAILS_ROOT/public`

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Abb.1: Flowchart der Blog Applikation

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

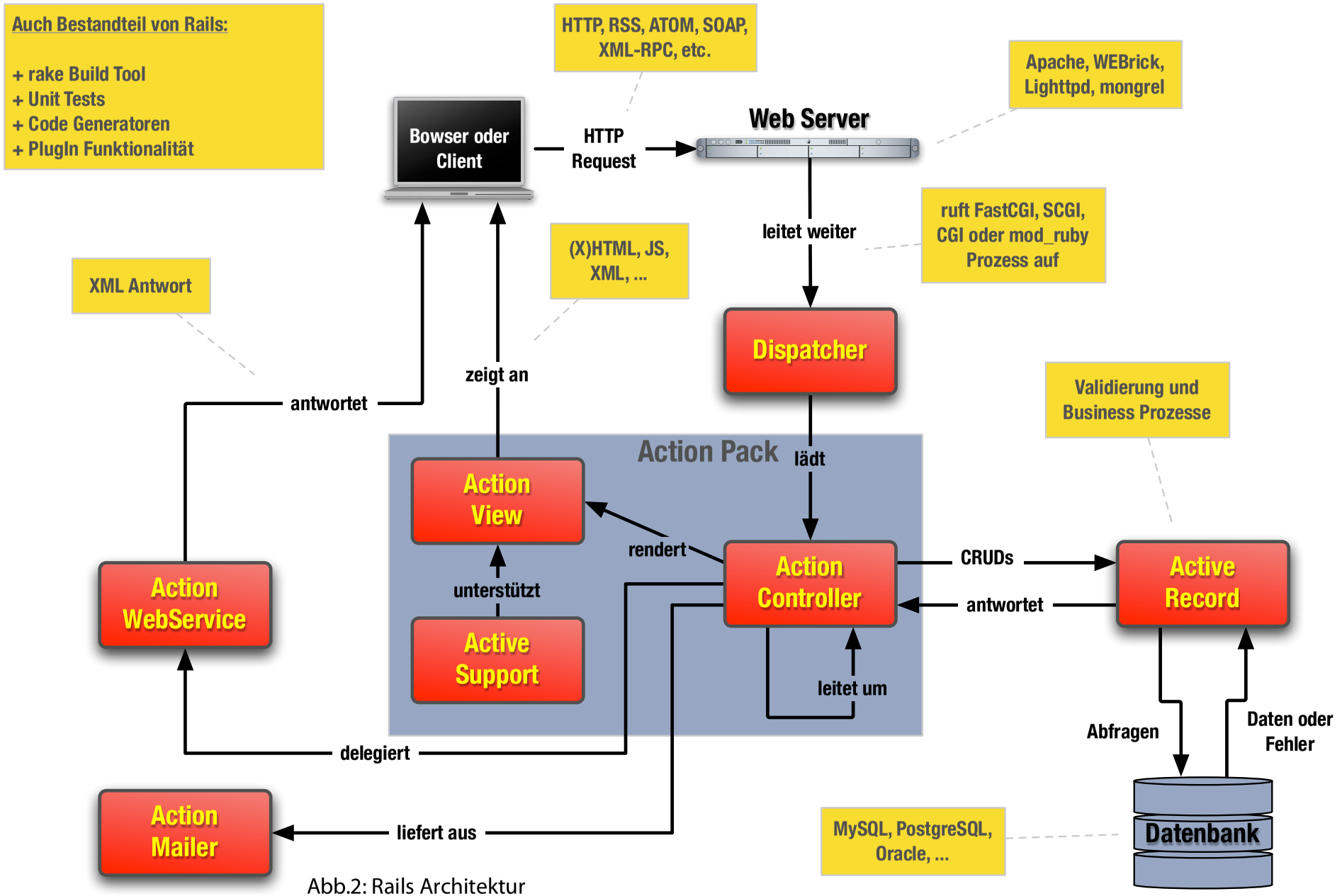


Abb.2: Rails Architektur

Methods

Strings
capitalize!
center
chomp!
chop!
concat
count
crypt
delete!
downcase!
dump
each
each_byte
empty?
gsub!
hash
hex
include?
index
intern
length
ljust, rjust
next!
oct
replace
reverse!
rindex
scan
slice!
split
squeeze!
strip!
sub!
sum
swapcase!
tr!
tr_s!
unpack
upcase!
upto

Regex
escape
last_match
new
quote
casefold?
kcode
match
source

Time
asctime
ctime
day
gmt?
gmtime
hour
isdst
localtime
mday
min
mon
month
sec
strftime
tv_sec
tv_usec
usec
utc
utc?
wday
yday
year
zone

DEFAULT DIRECTORY STRUCTURE

- rails_root
  - app
    - apis
    - controllers
      - application.rb
    - helpers
      - application_helper.rb
    - models
    - views
      - layouts
  - components
  - config
    - environments
      - development.rb
      - production.rb
      - test.rb
    - database.yml
    - environment.rb
    - routes.rb
  - db
  - doc
  - lib
  - log
    - development.log
    - production.log
    - server.log
    - test.log
  - public
    - images
    - javascripts
      - controls.js
      - dragdrop.js
      - effects.js
      - prototype.js
    - stylesheets
    - .htaccess
    - 404.html
    - 500.html
    - dispatch.cgi
    - dispatch.fcgi
    - dispatch.rb
    - favicon.ico
    - index.html
  - script
  - test
    - fixtures
    - functional
    - mocks
      - development
      - test
    - unit
    - test_helper.rb
  - vendor

METHODS NOTE

! - Denotes where a tailing ! may be used. A colourless ! denotes that the ! is compulsory.

PRE-DEFINED VARIABLES

$!	Exception information
$&	String of last match
$`	String left of last match
$'	String right of last match
$+	Last group of last match
$N	Nth group of last match
$=	Case insensitive flag
$/	Input record separator
$\	Output record separator
$,	Output field separator
$.	Current line number of last file read
$>	Default output for print
$_	Last input line of string
$0	Name of script
$*	Command line arguments
$stderr	Standard error output
$stdin	Standard input
$stdout	Standard output
$-a	True if -a is set.
$-d	Status of -d switch
$-l	True if -l is set
$-p	True if -p is set
$-v	Verbose Flag

RESERVED WORDS

=begin	elsif	rescue
=end	end	retry
BEGIN	ensure	return
END	false	self
alias	for	super
and	if	then
begin	in	true
break	module	undef
case	next	unless
class	nil	until
def	not	when
defined?	or	while
do	redo	yield
else		

REGULAR EXPRESSIONS SYNTAX

^	Start of string
$	End of string
.	Any single character
(a|b)	a or b
(...)	Group section
[abc]	Item in range (a or b or c)
[^abc]	Not in range (not a or b or c)
a?	Zero or one of a
a*	Zero or more of a
a+	One or more of a
a{3}	Exactly 3 of a
a{3,}	3 or more of a
a{3,6}	Between 3 and 6 of a
!(pattern)	"Not" prefix. Apply rule when URL does not match pattern.

Methods

Arrays
assoc
at
clear
collect!
compact!
concat
delete
delete_at
delete_if
each
each_index
empty?
eql?
fill
first
flatten!
include?
index
indexes
join
last
length
nitems
pack
pop
push
rassoc
reject!
replace
reverse!
reverse_each
rindex
shift
slice!
sort!
uniq!
unshift

Validation
condition_block?
create!
evaluate_condition
validate
validate_on_create
validate_on_update
validates_acceptance_of
validates_associated
validates_confirmation_of
validates_each
validates_exclusion_of
validates_format_of
validates_inclusion_of
validates_length_of
validates_numericality_of
validates_presence_of
validates_size_of
validates_uniqueness_of

Enumerable Mixin
collect
each_with_index
entries
find
find_all
grep
include?
max
min
reject
sort

Available free from ILoveJackDaniels.com

Ruby on Rails Logo used with permission.

Tag 1, Hands-On 3: M in MVC

Ziel des Hands-on	Die Fähigkeiten von ActiveRecord stärker nutzen
Aufgabe	1. Im vorherigen Hands-On ist die Tabelle für die Blogeinträge durch ein einfaches SQL Skript erzeugt worden. Allerdings ist diese Methode nicht besonders portabel gegenüber anderen Datenbanksystemen. Daher soll die Tabelle mit Hilfe von ActiveRecord::Migration angelegt werden. 2. Jeder Blogeintrag soll einen Zeitstempel erhalten, wann er angelegt wurde, und wann er das letzte Mal verändert worden ist. Dafür bietet ActiveRecord ein automatisches Feature. Die benötigten Spalten dazu sollen per Migration der Tabelle der Blogeinträge hinzugefügt werden. 3. Am Ende soll natürlich das Datum auch ausgegeben werden. Die entsprechenden Anpassungen sollen in den jeweiligen Views vorgenommen werden.
Ressourcen	http://wiki.rubyonrails.org/rails/pages/UnderstandingMigrations http://wiki.rubyonrails.org/rails/pages/UsingMigrations
Shortcuts	**Migration erzeugen**: `script/generate migration <name>` **Migrationsdateien**: `db/XXX_<name>.rb` **Wichtige Informationen:** • In der `self.down` Methode müssen immer die Aktionen der `self.up` Methode rückgängig gemacht werden. • Ausführen der Migration `rake db:migrate` **Wichtige Methoden:** • `create_table(table_name, &block) / drop_table(remove_table)` • `column(column_name, type) # Neue Spalte während des 'create_table' Blocks` • `add_column(table_name, column_name, type, options) / remove_column` **Bekannte Datentypen:** • Die Datentypen werden in die spezifischen DB Typen übersetzt. • integer, float, datetime, date, timestamp, time, text, string, binary, boolean **Automatische Spalten:** • `created_(at|on) / updated_(at|on)`. Mit `_at` wird ein kompletter Timestamp gespeichert, mit `_on` nur das Datum

Tag 1, Hands-On 4: M in MVC

Ziel des Hands-on	Validierung von Modelattributen
Aufgabe	1. Da gerade bei einem Blogsystem die Daten typischerweise aus einer Benutzereingabe in die Datenbank geschrieben werden, sollen diese Eingaben nicht ungesehen in der Datenbank landen. ActiveRecord bietet dafür sowohl fertige Validatoren an, als auch die Möglichkeit eigenen Validierungen zu definieren. Aufgabe ist es also die Felder eines Blogeintrags entsprechend zu validieren, dass keine unkorrekten oder gar bösartige Werte in der Datenbank landen. 2. Falls ein User eine nicht-valide Eingabe vornimmt soll er darüber informiert werden.
Ressourcen	http://api.rubyonrails.com/classes/ActiveRecord/Validations.html http://api.rubyonrails.com/classes/ActiveRecord/Validations/ClassMethods.html

Shortcuts	**Wichtige Informationen:**
	• Die Validierung findet statt bevor das Objekt gespeichert wird.
	• Jedes ActiveRecord Objekt hat eine `validate` Methode, mit der die Validierung explizit getriggert werden kann.
	• Scheitert eine Validierung, so wird die Fehlermeldung an das `errors` Hash des Objekts angehangen und kann in der View zur Fehlerausgabe verwendet werden.
	• Typischerweise verwendet man das `flash` Objekt zur Ausgabe solcher Meldungen (das `flash` Objekt ist ein Hash, durch das Scaffolding wissen die Views bereits, dass es einen Key `notice` im `flash` Hash geben kann).
	Einige Standardvalidierungen:
	• `validates_presence_of :name`
	• `validates_uniqueness_of :key`
	• `validates_format_of :email`
	• `validates_size_of :password`
	Eigene Validierung:
	`def validate`
	`  errors.add(birthday, “Who are you? Methusalem?”) unless`
	`  birthday < Time.parse(“01-01-1850”)`
	`end`

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

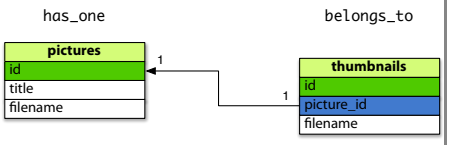
Tag 2, Hands-On 5: Relationen zwischen Modellen

Ziel des Hands-on	Relationen zwischen Modellen
Aufgabe	1. Neben dem Modell `Article` ist das Modell `Comment` für die Kommentare zu einem Artikel gedacht. Die beiden Modelle gehen eine Relation ein, die in Rails mittels `ActiveRecord::Associations` abgebildet werden sollen. Dabei sollen gegebenenfalls die Datenbank überarbeitet werden, um Relationen abbilden zu können. 2. Relationen sollten validiert werden, um Relationsvorschriften gerecht zu werden.
Ressourcen	http://api.rubyonrails.com/classes/ActiveRecord/Associations/ClassMethods.html http://api.rubyonrails.com/classes/ActiveRecord/Validations.html

Shortcuts	**Wichtige Informationen:**
	• Folgende Beziehungstypen (siehe nächste Seite für Beispiele)
	  • `has_one`
	  • 1:1 Beziehung auf Objektebene
	  • Erweitert Picture um die Methode thumbnail
	  • `belongs_to`
	  • Beschreibt die Tabelle, die den Foreign-Key in einer 1:n / 1:1 Beziehung hält
	  • Erweitert `Address` um die Methode `person`
	  • `has_many`
	  • 1:n Beziehung auf Objektebene
	  • Erweitert Person um die Methode `addresses`
	  • `has_many_and_belongs_to`
	  • n:m Beziehung auf Objektebene
	  • Zwischentabelle, die lediglich in der Datenbank abgebildet wird (keine ActiveRecord::Base Klasse).
	  • `has_many :through`
	  • n:m Beziehung
	  • Zwischentabelle wird durch `ActiveRecord::Base` Klasse definiert
	• Foreign-Keys müssen manuell angelegt werden
	  • Konvention Klassenname (Singular) + _id
	• Wichtige Informationen zur Validierung im Hands-On 4.

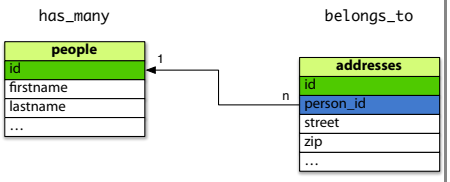
Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

has_one



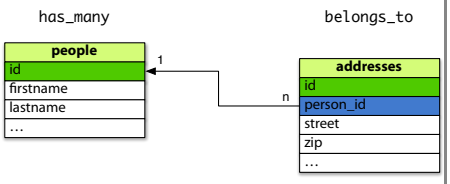

```
class Picture < ActiveRecord::Base
  has_one :thumbnail
end
class Thumbnail < ActiveRecord::Base
  belongs_to :picture
end
```

belongs_to



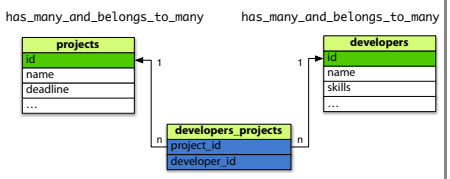

```
class Picture < ActiveRecord::Base
  has_one :thumbnail
end
class Thumbnail < ActiveRecord::Base
  belongs_to :picture
end
```

has_many



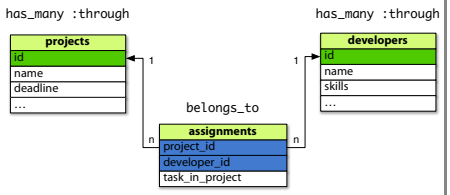

```
class Person < ActiveRecord::Base
  has_many :addresses
end
class Address < ActiveRecord::Base
  belongs_to :person
end
```

has_many_and_belongs_to




```
class Project < ActiveRecord::Base
  has_and_belongs_to_many :developers
end
class Developer < ActiveRecord::Base
  has_and_belongs_to_many :projects
end
```

has_many :through




```
class Project < ActiveRecord::Base
  has_many :developers, :through => assignments
end
class Assignment < ActiveRecord::Base
  belongs_to :developer
  belongs_to :project
end
class Developer < ActiveRecord::Base
  has_many :projects, :through => assignments
end
```


Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Hands-On 6: Controller und Views

Ziel des Hands-on	Controller und Views an die Relationen anpassen
Aufgabe	1. Die Relationen zwischen den Modellen haben Auswirkungen auf die Präsentations -und Controllerlogik. Deswegen müssen die Controller und View-Elemente angepasst werden. Aus der Einzelansicht eines Artikels sollte die Möglichkeit gegeben werden Kommentare zu einem Artikel anzuzeigen und neue Kommentare für diesen Artikel hinzuzufügen. 2. Die ActiveRecord Methoden `find` und `update` dienen als Abstraktionsmethoden für Datenbankoperationen. Es soll mittels der Methode `find` verschiedene Abfragen getätigt werden, die Auswirkungen auf die Artikel haben.
Ressourcen	http://api.rubyonrails.com/classes/ActionController/Base.html http://api.rubyonrails.com/classes/ActionView/Base.html http://ar.rubyonrails.org/classes/ActiveRecord/Base.html
Shortcuts	**Wichtige Informationen:** • Das Template zur Einzelansicht eines Artikels befindet sich in (`app/views/articles/show.rhml`) • Ein Template zur Auflisten alle Kommentare sollte implementiert werden. • Ein Template zum Anlegen eines Kommentar sollte implementiert werden. • Der Zugriff der Kommentar-Templates sollte über die Einzelansicht eines Artikels erfolgen. • Die zugehörigen Actions sollten sinnvoll benannt und an geeigneter Stelle implementiert werden. **Wichtige Methoden im Controller:** • Werteübergabe mittels `params[:name]` • `render :template => 'name'` • `redirect_to :action => 'name'` **Wichtige Methoden in der View:** • `<% form_for :comment, @comment do |f| %>` • `<%= f.text_field :author %>` • `<%= submit_tag 'Save' %>` • `<%= link_to 'Edit', :action => 'edit', :id => @article %>` • `<%= h @article.title %>`

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Ziel des Hands-on	Die ActiveRecord Methode `find` und `update` näher kennen lernen

Shortcuts

Wichtige Informationen zur Aufgabe:

- Auf der Startseite des Blogs werden bestimmte Artikel angezeigt.
  - Es sollen die letzten 10 aktuellsten Artikel in der Liste ausgegeben werden.
  - Es sollen alle Artikel vom **heutigen Tag** ausgegeben werden. (Link auf der Startseite „Artikel von Heute“)
  - Es sollen alle Artikel vom **gestrigen Tag** ausgegeben werden. (Link auf der Startseite „Artikel von Gestern“)
  - Es sollen alle Artikel der **die älter sind** ausgegeben werden. (Link auf der Startseite „Artikel von letzter Woche“)

Wichtige Informationen zu ActiveRecord find:

- **Find by id**
  - ```
    Person.find(1)
    # returns the object for ID = 1
    ```
  - ```
    Person.find(1, 2, 6)
    # returns an array for objects with IDs in (1, 2, 6)
    ```
  - ```
    Person.find([7, 17])
    # returns an array for objects with IDs in (7, 17)
    ```
  - ```
    Person.find([1])
    # returns an array for objects the object with ID = 1
    ```
  - ```
    Person.find(1, :conditions => "administrator = 1",
    :order => "created_on DESC")
    ```
- **Find first**
  - ```
    Person.find(:first)
    # returns the first object fetched by
    SELECT * FROM people
    ```
  - ```
    Person.find(:first,
    :conditions => [ "user_name = ?", user_name])
    ```
  - ```
    Person.find(:first, :order => "created_on DESC",:offset => 5)
    ```
- **Find all**
  - ```
    Person.find(:all)
    # returns an array of objects for all the rows fetched by
    SELECT * FROM people
    ```
  - ```
    Person.find(:all, :conditions => [ "category IN (?)",
    categories], :limit => 50)
    ```
  - ```
    Person.find(:all, :offset => 10, :limit => 10)
    ```
  - ```
    Person.find(:all, :include => [ :account, :friends ])
    ```
  - ```
    Person.find(:all, :group => "category")
    ```
    ```

Hands-On 7: Partials

Ziel des Hands-on	Liste der Kommentare als Partials implementieren
Aufgabe	1. Die Liste der Kommentare eines Artikels soll als Partial eingebunden werden. Dabei ist es wichtig das „Partial"-Konzept zu verstehen. Um Wiederholungen bei der Ausgabe der Kommentare zu vermeiden, sollte das Partial mittels einer Collection eingebunden werden. 2. Gemäß dem DRY-Prinzip sind Partials unerlässlich. Es sollen weitere Möglichkeiten zur Erstellung von Partials gefunden und implemenitert werden.
Ressourcen	http://api.rubyonrails.com/classes/ActionView/Partials.html

Shortcuts	**Wichtige Informationen zur Nutzung von Partials:**
	• Partial mit Übergabe einer lokalen Variable
	`<%= render :partial => "person" %>`
	`(Partial _comment.rhtml, lokale Variable person)`
	`<%= render :partial => "account",`
	`:locals => { :account => @customer.account } %>`
	`(Partial _account.rhtml, lokale Variable account)`
	• Partial mit Übergabe einer Collection
	(Iteration z.B. über ein Array, Ausgabe für jedes Objekt im Array)
	`<%= render :partial => "comment",`
	`:collection => @articles.comments %>`
	`(Partial _comment.rhtml, lokale Variable comment)`
	• Partials mit anderen Controllern teilen
	`<%= render :partial => "user/login",`
	`:locals => { :user => @user.id } %>`
	`(Partial /user/_login.rhtml, lokale Variable user)`

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Hands-On 8: Tagging

Ziel des Hands-on	Artikel im Blog sollen mit Tags versehen werden
Aufgabe	1. Fast jeder Blog bietet die Funktionalität Schlagwörter, sog. Tags einem Artikel hinzuzufügen. Es können mehrere Tags zu einem Artikel hinzugefügt werden. Ein Tag kann auch mehreren Artikeln zugeordnet werden. 2. Die Relation zwischen Artikel und Tag sollte in Rails modelliert werden. 3. Beim Erstellen und Bearbeiten eines Artikels sollte es die Möglichkeit geben vorhandene Tag bzw. neue Tags einem Artikel hinzuzufügen. 4. Die Tags sollten bei der Anzeige des Artikels ebenfalls angezeigt werden.
Ressourcen	Bisherige Hands-Ons Kreatives Engagement

Fachhochschule Köln
Cologne University of Applied Sciences
Fakultät für Informatik und Ingenieurwissenschaften

Hands-On 9: User-Login und Testen in Rails

Ziel des Hands-on	User-Login und Testen in Rails
Aufgabe	1. Ein User-Login im Blog-System dient dazu, dass Besucher keine Artikel anlegen können. Schliesslich soll der Blog ein privater Blog bleiben. Ziel ist es ein User-Login in Rails zu implementieren. 2. Zum Testen der Login-Funktionalität sollen Tests geschrieben werden. Im ersten Schritt soll die Rails-Umgebung kennen gelernt werden und der Unterschied zwischen Functional- und Unit-Tests ersichtlich sein.
Ressourcen	http://api.rubyonrails.com Keywords: ActionController::Assertions Ruby Test Cheat Sheet

Shortcuts	**Wichtige Informationen zum Login:**
	• `before_filter :authenticated, :only => [:new, :create]`
	• Authentisierungsmechanismus im Model implementiert
	• Passwörter sollten nicht im Klartext in der DB stehen (Hashfunktion: Klasse `Digest`)
	• User steht nach Login in der Session, Session Zugriff erfolgt über `session[:some_key]`
	• `authenticated` Methode im Application Controller
	Wichtige Informationen zum Testen:
	• Die Test-Umgebung befindet sich im Ordner `test`
	• Die Unterordner werden wie folgt beschrieben
	  • `fixtures` (Testdaten im YAML-Dateien)
	  • `functional` (Funktionale Testdaten -Controller)
	  • `integration` (Systemweite Testdaten)
	  • `mocks` (Mockup-Objekte)
	  • `unit` (Testdaten für Models)
	• `rake test` ermöglicht den Durchlauf aller Tests.
	• Assertions auf dem Cheat Sheet

Ruby on Rails Testing

Standard Assertions

```
assert                  boolean
assert_equal            expected, actual
assert_raise            *args
assert_raises           *args, &block
assert_instance_of      klass, object
assert_nil              object
assert_kind_of          klass, object
assert_respond_to       object, method
assert_match            pattern, string
assert_same             expected, actual
assert_operator         object1, operator, object2
assert_nothing_raised   *args
assert_not_same         expected, actual
assert_not_equal        expected, actual
assert_not_nil          object
assert_no_match         regexp, string
assert_throws           expected_symbol, &proc
assert_nothing_thrown   &proc
assert_in_delta         expected_float, actual_float, delta
assert_send             send_array
```

Rails Assertions

```
assert_response         type
assert_redirected_to    options = {}
assert_template         expected

assert_recognizes       expected_options, path, extras={}
assert_generates        expected_path, options, defaults={},
                             extras = {}
assert_routing          path, options, defaults={}, extras={}

assert_tag              *opts
assert_no_tag           *opts
assert_dom_equal        expected, actual
assert_dom_not_equal    expected, actual

assert_valid            record
```

Test::Rails Assertions

```
assert_assigned         ivar, value = NOTHING
• deny_assigned
assert_content_type     type, message = nil
assert_flash            key, content

assert_image            src

assert_error_on         field, type
assert_field            form_action, type, model, column, value = nil
assert_input            form_action, type, name, value = nil
assert_label            form_action, name, include_f = true
assert_links_to         href, content = nil
• deny_links_to
assert_multipart_form   form_action
assert_post_form        form_action
assert_select           form_action, model, column, options
assert_submit           form_action, value
assert_tag_in_form      form_action, options

• deny
assert_empty            obj
• deny_empty
assert_includes         obj, item, message = nil
• deny_includes
```

Most also take a message argument as the last parameter. The message will be shown if the test fails.

See also PeepCode.com Screencasts and the upcoming Topfunky Field Guide to Ruby on Rails

Hands-On 10: AJAX in Rails

Ziel des Hands-on	AJAX kennen lernen undin Rails anwenden können
Aufgabe	1. AJAX ist das „Polishing“ für Web-Applikationen. Rails bietet dabei einfache Möglichkeiten AJAX-Funktionen zu nutzen. 2. Neben den AJAX Helper Methoden bietet Rails mit den Javascript Generator Templates, die Möglichkeit Zustandsänderungen von Templates zu bestimmen. Diese Möglichkeiten sollten bei der Entwicklung von Rails Web-Applikationen berücksichtigt werden. 3. Ziel des Hands-On ist es eine Live-Suche für Artikel mit den gegebenen Rails Funktionen zu implementieren. Da bei AJAX Anwendungen Aktionen im Hintergrund ablaufen, muss der User bei nicht sichtbaren Aktionen visuell benachrichtigt werden (Eieruhr-Prinzip).
Ressourcen	http://api.rubyonrails.org/classes/ActionView/Helpers/PrototypeHelper.html http://api.rubyonrails.org/classes/ActionView/Helpers/ScriptaculousHelper.html http://api.rubyonrails.org/classes/ActionView/Helpers/PrototypeHelper/JavaScriptGenerator/GeneratorMethods.html

Shortcuts	**Javascript Generator Templates (Controller)**
	
	`render :update do |page|`
	`  page.insert_html :bottom, 'list', "<li>#{@item.name}</li>"`
	`  replace_html 'result', :partial => 'search_result',`
	`  :object => @article`
	`  page.visual_effect :highlight, 'result'`
	`  page.hide 'status-indicator', 'dom-element'`
	`end`
	
	Protoype/script.aculo.us Wrapper-Methoden (View)
	
	• `form_remote_tag :html => { :action =>`
	`url_for(:controller => "some", :action => "place") }`
	
	• `link_to_remote "Destroy",`
	`:url => person_url(:id => person), :method => :delete`
	
	• `observe_field (siehe Vortragsfolien)`
	
	• `draggable_element("my_image", :revert => true)`
	
	• `sortable_element("mylist", :url => {:action => "order" })`