

Hussein Morsy, Tanja Otto

Ruby on Rails 2

Das Entwickler-Handbuch

Auf einen Blick

TEIL I: Grundlagen

1	Einführung	27
2	Installation	41
3	Erste Schritte	59
4	Einführung in Ruby	67

TEIL II: Beispiel-Applikationen

5	Eine einfache Bookmarkverwaltung	153
6	Test-Driven Development	241

TEIL III: Das Rails-Framework

7	Rails-Projekte erstellen	289
8	Templatesystem mit ActionView	325
9	Steuerzentrale mit ActionController	399
10	Datenbankzugriff mit ActiveRecord	425
11	E-Mails verwalten mit ActionMailer	523
12	Nützliche Helfer mit ActiveSupport	539
13	Ajax on Rails	551
14	RESTful Rails und Webservices	569

TEIL IV: Rails im Einsatz

15	Rails mit Plug-ins erweitern	593
16	Performancesteigerung	619
17	Sicherheit	647
18	Veröffentlichen einer Rails-Applikation auf einem Server	657

Inhalt

Geleitwort des Fachgutachters	19
Einleitung	21
Für wen wurde dieses Buch geschrieben?	21
Was befindet sich in diesem Buch?	21
Hinweise zu Listings	22
Was ist auf der CD-ROM	22
Die Website zum Buch	22
Wie wir dieses Buch erstellt haben	23
Danke	23

TEIL I: GRUNDLAGEN

1 Einführung 27

1.1	Wie entstand Rails?	27
1.2	Warum Ruby?	28
1.3	Model View Controller	30
1.4	Datenbankpersistenz	31
1.5	Konvention statt Konfiguration	31
1.6	Das DRY-Prinzip	32
1.7	Das neue Web 2.0	32
1.8	Neues in Rails 2.0.x	33
1.9	Update auf Rails 2.0.x	36

2 Installation 41

2.1	Allgemeines	41
2.2	Installation unter Mac OS X	42
2.2.1	Updaten von Komponenten unter Leopard	43
2.2.2	Installation von MySQL unter Leopard	44
2.2.3	Installation von Ruby on Rails mit MacPorts	45
2.3	Installation unter Windows	48
2.3.1	Installation von Instant Rails	48
2.3.2	Der Instant Rails Manager	49
2.3.3	Eine Rails-Applikation erstellen und starten	49
2.3.4	Datenbanken verwalten mit phpMyAdmin	52
2.3.5	Gem-Pakete verwalten	52
2.3.6	Generatoren	53

2.4	Installation unter Linux	53
2.4.1	Installation von MySQL	53
2.4.2	Installation von SQLite3	54
2.4.3	Installation von PostgreSQL	54
2.4.4	Installation von Ruby	54
2.4.5	Installation von RubyGems	54
2.4.6	Installation von Rails und anderen Gem-Paketen	55
2.5	Editoren und Entwicklungsumgebungen	55
2.5.1	TextMate	55
2.5.2	E-Text-Editor	56
2.5.3	Vim	56
2.5.4	Emacs	56
2.5.5	Aptana	56
2.5.6	NetBeans	57
2.5.7	IntelliJ IDEA	57
2.5.8	CodeGear 3rdRail	58
2.5.9	Visual Studio	58
3	Erste Schritte	59
3.1	Eine Rails-Applikation erstellen	59
3.2	Der lokale Rails-Server	60
3.3	Grundgerüst mit Scaffolds erstellen	61
3.4	Die Applikation im Browser aufrufen	63
3.5	HTTP-Authentifizierung	64
4	Einführung in Ruby	67
4.1	Was ist Ruby?	67
4.2	Ruby-Code ausführen	68
4.2.1	Quelltext	69
4.2.2	Interaktive Ruby Shell – irb	69
4.2.3	Im Webbrowser Try Ruby	70
4.3	Grundlagen	71
4.3.1	Variablen	72
4.3.2	Objekte und Datentypen	73
4.3.3	Verzweigungen	78
4.3.4	Überprüfen auf Gleichheit	81
4.3.5	Schleifen	82
4.3.6	Klassen	84

4.4	Zahlen	88
4.5	Zeichenketten	91
4.5.1	here-document	93
4.5.2	Ausdrücke in Zeichenketten	94
4.5.3	Die Methode length	95
4.5.4	Die Methode split	96
4.5.5	Zeichenketten formatieren	97
4.5.6	Groß- und Kleinschrift	98
4.5.7	Teilstrings	99
4.5.8	In Zeichenketten suchen	101
4.5.9	Etwas einer Zeichenkette hinzufügen	102
4.5.10	Angehängte Zeichen löschen	103
4.5.11	Leerräume löschen	104
4.5.12	Zeichenketten wiederholen	104
4.5.13	Komma-separierte Daten analysieren	104
4.5.14	Strings in Zahlen konvertieren	105
4.5.15	Zeichenketten verschlüsseln	107
4.5.16	Zeichen in einer Zeichenkette zählen	108
4.5.17	Eine Zeichenkette umkehren	108
4.5.18	Doppelte Zeichen entfernen	109
4.5.19	Bestimmte Zeichen entfernen	109
4.5.20	Leerräume drucken	110
4.6	Symbole	110
4.7	Reguläre Ausdrücke	111
4.7.1	Syntax von Regulären Ausdrücken	111
4.7.2	Anwendungsbeispiele aus der Praxis	113
4.8	Arrays	116
4.8.1	Ein Array erzeugen	116
4.8.2	Auf Arrayelemente zugreifen	117
4.8.3	Auf die Länge eines Arrays zugreifen	120
4.8.4	Arrays vergleichen	120
4.8.5	Ein Array sortieren	121
4.8.6	Ein Array zufällig sortieren	123
4.8.7	Nach Elementen in einem Array suchen	123
4.8.8	Differenz zwischen zwei Arrays bestimmen	125
4.8.9	nil-Werte aus einem Array entfernen	126
4.8.10	Bestimmte Array-Elemente entfernen	126
4.8.11	Ein Array umkehren	127
4.8.12	Doppelte Einträge aus einem Array löschen	128
4.8.13	Iteratoren	128

4.9	Hashes	130
4.9.1	Einen Hash erzeugen	130
4.9.2	Zugriff auf Elemente eines Hashs	132
4.9.3	Schlüssel-Wert-Paare löschen	134
4.9.4	Über einen Hash iterieren	135
4.9.5	Schlüssel und Wert in einem Hash vertauschen .	136
4.9.6	Schlüssel und Werte in einem Hash finden	136
4.9.7	Einen Hash in ein Array extrahieren	137
4.9.8	Nach Schlüssel-Wert-Paaren suchen	138
4.9.9	Einen Hash sortieren	138
4.9.10	Zwei Hashes miteinander mischen	139
4.9.11	Einen Hash aus einem Array erzeugen	139
4.10	Datum und Zeit	139
4.10.1	Die aktuelle Zeit bestimmen	140
4.10.2	Mit bestimmten Zeiten arbeiten	140
4.10.3	Einen Wochentag bestimmen	141
4.10.4	Mit der Unix-Zeit arbeiten	142
4.10.5	Einen Tag im Jahr ermitteln	142
4.10.6	Wochenzahlen ermitteln	143
4.10.7	Schaltjahre bestimmen	144
4.10.8	Uhrzeiten ausgeben	144
4.10.9	Datums- und Zeitwerte vergleichen	145
4.10.10	Mit Datums- und Zeitwerten rechnen	145
4.10.11	Datums- und Zeitwerte aus Zeichenketten ermitteln	146
4.11	Module	147
4.11.1	Namensräume	147
4.11.2	Mixins	148

TEIL II: BEISPIEL-APPLIKATIONEN

5 Eine einfache Bookmarkverwaltung **153**

5.1	Rails-Projekt erstellen	154
5.1.1	Erstellung des Bookmarks-Controllers	156
5.1.2	View erstellen	159
5.2	Weitere Views anlegen	163
5.3	Layout	165
5.4	Model	171
5.4.1	Migrations	173
5.4.2	ActiveRecord	175

5.4.3	Datenbankzugriff in der Konsole testen	176
5.5	CRUD – Create – Read – Update – Delete	180
5.6	Fehlerbehandlung in Formularen	192
5.7	Flash-Messages	197
5.8	Refaktorisierung mit Helper und Partial	200
5.8.1	Helper	200
5.8.2	Partials	203
5.9	Authentifizierung	206
5.10	Routing	216
5.10.1	Benannte Routen	220
5.10.2	Die root-Route	221
5.11	RESTful Rails	223
5.12	Darstellungsformate in RESTful Rails	230
5.13	Ajax	232
5.13.1	Einbinden von JavaScript-Bibliotheken für Ajax ..	233
5.13.2	Ajaxbasierte Formulare	234
5.13.3	Ajax im Controller	235
5.13.4	RJS-Templates	237
6	Test-Driven Development	241
6.1	Was ist TDD?	241
6.2	Vorstellung des Projekts	243
6.3	Projekt erstellen und konfigurieren	245
6.4	Unit-Tests	246
6.4.1	Erstellung des Country-Models	246
6.4.2	Erstellung des Airport-Models	257
6.4.3	Erstellung des Flight-Models	264
6.5	Functional-Tests erstellen	269
6.5.1	Der Functional-Test des Flights-Controllers	270
6.5.2	Fixtures in die Entwicklungsdatenbank laden	274
6.5.3	Anzeigen der Flughafen-Codes	275
6.5.4	Datumswerte formatieren	276
6.5.5	Select-Felder zur Flughafenauswahl	277
6.5.6	Weitere Funktionen zur Übung	279
6.5.7	Integration-/Acceptance-Tests erstellen	279
6.5.8	Integration-Test mit Rails	280
6.6	Autotest	282
6.6.1	Installation	282
6.6.2	Verwendung	282
6.7	Referenz	283

6.7.1	Rake-Tasks für Tests	283
6.7.2	Die wichtigsten Assert-Befehle	283

TEIL III: DAS RAILS-FRAMEWORK

7 Rails-Projekte erstellen 289

7.1	Generieren eines Rails-Projektes	289
7.1.1	Datenbank festlegen	289
7.1.2	Freeze Rails	290
7.1.3	Rails-Befehl auf vorhandenem Projekt anwenden	291
7.1.4	Sonstige Optionen	291
7.2	Verzeichnisstruktur einer Rails-Applikation	292
7.3	Namenskonventionen	294
7.4	Datenbank-Konfiguration	296
7.5	Umgebungseinstellungen	299
7.5.1	Entwicklungsumgebung (development)	300
7.5.2	Produktionsumgebung (production)	300
7.5.3	Testumgebung (test)	302
7.5.4	Eigene Umgebungen anlegen	303
7.5.5	Clean up your environment	303
7.6	Generatoren	304
7.6.1	Verwendung	304
7.6.2	Übersicht aller Generatoren	305
7.6.3	Rückgängig machen	307
7.7	Rails-Konsole	308
7.8	Lokaler Server	309
7.9	Logging	310
7.10	Debugging	311
7.10.1	Installation von ruby-debug	311
7.10.2	Breakpoint setzen	312
7.10.3	Server mit Debugger starten	312
7.10.4	Debugger-Konsole	313
7.10.5	Ausführung fortsetzen	313
7.11	Rake	314
7.12	EdgeRails	319
7.13	Ein Rails-Projekt in Subversion überführen	319
7.13.1	Ein Rails-Projekt in Subversion importieren	320
7.13.2	Ein Rails-Projekt aus Subversion auschecken	322
7.13.3	Ignorieren von Dateien	322

8 Templatesystem mit ActionView 325

8.1	ERB-Templates	326
8.2	Erstellung von Templates	328
8.3	Helper	329
8.3.1	Helper für Verlinkungen	330
8.3.2	Helper zum Einbinden von Stylesheets und JavaScripts	337
8.3.3	Helper zur Zahlenformatierung	339
8.3.4	Helper zur Textmanipulation	342
8.3.5	Helper zur Textformatierung	345
8.3.6	Helper zur Entfernung von HTML-Code	346
8.3.7	Sonstige Helper	347
8.3.8	Eigene Helper entwickeln	350
8.4	Layouts	351
8.5	Formulare	353
8.5.1	Formulare mit Bezug zu einem Model	353
8.5.2	Validierung	379
8.5.3	Formulare mit Bezug zu mehr als einem Model .	385
8.5.4	Formulare ohne Bezug zu einem Model	387
8.6	Partials	389
8.6.1	Übergabe von Parametern mit :locals	391
8.6.2	Partials mit Ressourcen	392
8.6.3	Shared Partials	394
8.6.4	Layout-Partials	396
8.7	Alternative Template-Systeme	396

9 Steuerzentrale mit ActionController 399

9.1	Grundlagen	399
9.2	Aufgaben des Controllers	401
9.2.1	Daten aus HTTP-Anfragen empfangen	401
9.2.2	Datenbankabfragen über Model-Klassen	403
9.2.3	Setzen und Abfragen von Cookies	404
9.2.4	Setzen und Abfragen von Sessions	405
9.2.5	Templates aufrufen	405
9.2.6	Setzen von Flash-Nachrichten	408
9.2.7	Weiterleitungen	409
9.2.8	Senden von Dateien und Daten	411
9.2.9	Authentifizierung	413
9.3	Filter	414

9.3.1	Filtertypen	414
9.3.2	Filter nur auf bestimmte Actions anwenden	416
9.4	Routing	417
9.4.1	Das Standard-Routing	417
9.4.2	Routing-Regeln erstellen	419
9.4.3	Benannte Routen	420
9.4.4	Die root-Route	420
9.4.5	Komplexe Routings mit Regulären Ausdrücken .	422
9.4.6	Routing mit Angabe der HTTP-Methode	423
9.4.7	Ressourcen	424

10 Datenbankzugriff mit ActiveRecord 425

10.1	Einführung	425
10.1.1	Vor- und Nachteile	427
10.1.2	Unterstützte Datenbanksysteme	428
10.1.3	Ein erstes Beispiel	429
10.1.4	Tabelle erstellen	431
10.2	Eine ActiveRecord-Model-Klasse generieren	435
10.2.1	Generieren eines Models mit dem Model-Generator	437
10.3	Rake-Tasks zum Verwalten von Datenbanken	440
10.3.1	Erstellen und Löschen von Datenbanken	440
10.3.2	Informationen über die Datenbanken	440
10.3.3	Migrations	441
10.3.4	Fixtures	442
10.3.5	Testdatenbank	442
10.3.6	Schema-Dateien	443
10.3.7	Sessions	443
10.4	Getter- und Setter-Methoden	444
10.4.1	Überschreiben der Getter- und Setter-Methoden	445
10.4.2	Automatische Typ-Erkennung	445
10.5	Erstellen, bearbeiten und löschen	446
10.5.1	Neues ActiveRecord-Objekt erstellen	446
10.5.2	ActiveRecord-Objekt erstellen und direkt speichern	447
10.5.3	Aktualisieren von Objekten	447
10.5.4	Löschen von Objekten	448
10.6	Suchen	449
10.6.1	Suche nach IDs	450

10.6.2	Suche nach dem ersten Datensatz oder Ausgabe aller Datensätze	451
10.6.3	Suchoptionen im Überblick	452
10.6.4	Suchbedingung (:conditions)	453
10.6.5	Sortierreihenfolge (:order)	455
10.6.6	Limitieren der Suchergebnisse	456
10.6.7	Suche mit dynamischen Find-Methoden	457
10.6.8	Suche über SQL	458
10.7	Datenbankschema und Migrations	459
10.8	Migration-Skripte	460
10.8.1	Migration-Skripte generieren	461
10.8.2	Änderungen ausführen	466
10.8.3	Änderungen rückgängig machen	467
10.9	Migration-Befehle im Detail	467
10.9.1	Datentypen	467
10.9.2	Tabellenfelder verwalten	469
10.9.3	Tabellen verwalten	470
10.9.4	Indizes verwalten	472
10.9.5	Beispieldaten hinzufügen	473
10.9.6	SQL-Befehle direkt verwenden	473
10.9.7	Schnappschuss eines Datenbankschemas	474
10.10	Assoziationen	475
10.10.1	Eins-zu-viele-Assoziationen (1:n)	475
10.10.2	Eins-zu-eins-Assoziationen (1:1)	484
10.10.3	Viele-zu-viele-Assoziationen (n:m)	487
10.10.4	Polymorphe Assoziationen	494
10.10.5	Mehrere Assoziationen zum gleichen Model	498
10.10.6	Assoziationen mit Bedingungen	499
10.10.7	Eine Assoziation um eigene Methoden erweitern	502
10.11	Validierung	503
10.11.1	validates_acceptance_of	503
10.11.2	validation_associated	504
10.11.3	validates_confirmation_of	505
10.11.4	validates_exclusion_of	506
10.11.5	validates_inclusion_of	507
10.11.6	validates_format_of	508
10.11.7	validates_length_of/validates_size_of	508
10.11.8	validates_numericality_of	511
10.11.9	validates_presence_of	512
10.11.10	validates_uniqueness_of	513

10.11.11	validates_each	514
10.12	Statistische Berechnungen	515
10.13	Callbacks	516
10.14	Vererbung	518
11	E-Mails verwalten mit ActionMailer	523
11.1	Beispielprojekt: Kontaktformular	523
11.2	HTML-E-Mails	535
11.3	E-Mails mit Anhängen	536
11.4	Konfiguration	536
12	Nützliche Helfer mit ActiveSupport	539
12.1	Zahlen	540
12.1.1	Gerade und ungerade Zahlen	540
12.1.2	Ordinalzahlen	540
12.1.3	Kapazitätseinheiten	541
12.1.4	Datum und Zeit	541
12.2	Zeichenketten	545
12.3	Arrays	546
12.4	Hashes	547
12.5	Datentypunabhängig	548
13	Ajax on Rails	551
13.1	Grundlagen	551
13.2	JavaScript-Frameworks	552
13.2.1	Prototype	552
13.2.2	Script.aculo.us	552
13.2.3	Weitere Frameworks	553
13.3	Einbinden der JavaScript-Bibliotheken	553
13.4	RJS: Ruby-JavaScript	555
13.4.1	RJS-Befehle einfügen	556
13.4.2	Aufrufen von RJS	557
13.4.3	Beispiel	559
13.5	Debugging mit Firebug	561
13.6	RJS-Referenz	564
13.6.1	Methoden zum Ein- und Ausblenden von Elementen	564

13.6.2	Methoden zum Hinzufügen, Ersetzen und Löschen von Elementen	565
13.6.3	Sonstige Methoden	567

14 RESTful Rails und Webservices 569

14.1	Was sind Webservices?	569
14.2	REST	570
14.2.1	Ressourcen / Nomen	571
14.2.2	HTTP-Methoden / Verben	572
14.3	RESTful Rails	573
14.3.1	Generatoren	573
14.3.2	Ressourcen	574
14.3.3	Verschachtelte Ressourcen	576
14.3.4	Namespaces	577
14.3.5	Singleton-Ressourcen	579
14.3.6	Ressourcen erweitern	582
14.4	Einen Webservice anbieten	584
14.4.1	Die Weboberfläche	584
14.4.2	Die API	585
14.5	Zugriff auf einen Webservice mit ActiveResource	587

TEIL IV: RAILS IM EINSATZ

15 Rails mit Plug-ins erweitern 593

15.1	Plug-ins Grundlagen	593
15.1.1	Plug-ins installieren und deinstallieren	594
15.1.2	Plug-ins-Quellen verwalten	596
15.2	Nützliche Plug-ins	597
15.2.1	Authentifizierung mit Restful Authentication	597
15.2.2	Mehrsprachige Applikationen mit Globalite	604
15.2.3	User-Fotos anzeigen mit Gravatar	614
15.2.4	Fehlerbenachrichtigung mit Exception Notifier ..	615
15.2.5	Weitere nützliche Plug-ins	616

16 Performancesteigerung 619

16.1	Einführung	619
16.2	Page-Caching	620
16.2.1	Grundlagen	620

- 16.2.2 Caching im Controller aktivieren 622
- 16.2.3 Einstellungen 622
- 16.2.4 Beispiel 623
- 16.2.5 Löschen von Cache-Dateien 624
- 16.2.6 Cachen der Root-Page 629
- 16.3 Action-Caching 631
 - 16.3.1 Grundlagen 631
 - 16.3.2 Beispiel 632
 - 16.3.3 Löschen von Cache-Dateien 634
 - 16.3.4 Verschiedene Caching-Strategien mischen 635
- 16.4 Fragment-Caching 635
 - 16.4.1 Grundlagen 635
 - 16.4.2 Beispiel 636
 - 16.4.3 Löschen von Cache-Dateien 638
- 16.5 Caching von CSS- und JavaScript-Dateien 640
 - 16.5.1 Grundlagen 640
 - 16.5.2 Beispiel 640
 - 16.5.3 Löschen der Cache-Dateien 642
- 16.6 Caching mit memcached 642
 - 16.6.1 Grundlagen 642
 - 16.6.2 Installation 644
 - 16.6.3 Installation des memcached-Servers 644
 - 16.6.4 Verwendung 645
 - 16.6.5 Löschen des Caches 646
- 16.7 Zusammenfassung 646

17 Sicherheit 647

- 17.1 Warum Sicherheit wichtig ist 647
- 17.2 SQL Injection 647
- 17.3 Cross-Site-Scripting XSS 648
- 17.4 Cross-Site Request Forgery (CSRF/XSRF) 650
- 17.5 Sessions 652
 - 17.5.1 Session Hijacking 652
 - 17.5.2 Session Fixation 652
- 17.6 Validierung 654
- 17.7 Reguläre Ausdrücke 655

18	Veröffentlichen einer Rails-Applikation auf einem Server	657
18.1	Wahl des Providers	658
18.2	Einrichten des Servers	659
18.2.1	Wahl der Server-Software	659
18.2.2	Deploy-User anlegen	660
18.2.3	Paketmanagement konfigurieren	661
18.2.4	Installation von Subversion und Erstellung eines Repository	663
18.2.5	Installation von Ruby und Co.	663
18.2.6	MySQL	664
18.2.7	Datenbank für das Beispielprojekt anlegen	664
18.2.8	Installation von SQLite3	665
18.2.9	Installation von Rails	665
18.2.10	Installation und Konfiguration von Mongrel	666
18.2.11	Apache-Webserver	667
18.2.12	Module aktivieren	668
18.3	Konfigurieren der Rails-Applikation auf dem Server	669
18.3.1	Konfiguration	669
18.3.2	Verzeichnis für Railsprojekt erstellen	672
18.3.3	Statische Seite testen	672
18.3.4	Mongrel-Cluster-Konfiguration vorbereiten	672
18.4	Deployment mit Capistrano	673
18.4.1	Konfiguration der Datenbank	673
18.4.2	Installation	674
18.4.3	Capistrano in Rails-Applikation konfigurieren	674
18.4.4	Verzeichnisse auf dem Server erstellen	675
18.4.5	Mongrel-Cluster-Konfigurationsdatei auf dem Server generieren	675
18.4.6	Übertragen und Starten der Rails-Applikation ...	676
18.4.7	Die letzte Übertragung rückgängig machen	677
18.4.8	Anzeigen einer Wartungsseite	677
18.4.9	Übertragung einzelner Dateien	680
18.4.10	Wie Capistrano einen Deploy durchführt	681
18.4.11	Verzeichnisse, die nicht im Repository sind, verwalten	684
18.4.12	Weitere Capistrano-Tasks	686
	Index	689

Geleitwort des Fachgutachters

»Hast du dieses Rails schon mal gesehen? ... musst du mal ausprobieren«. So oder so ähnlich fängt es bei den meisten an.

Bei mir war es nicht anders. Ein Kollege kam eines Tages auf mich zu und erzählte mir von einer neuen Programmiersprache, entwickelt von einem Japaner. Bisher sei die Dokumentation wohl nur auf Japanisch erhältlich, aber ein Amerikaner hätte endlich ein »nicht-japanisches« Buch dazu rausgebracht. Einige Zeit später tauchte auch noch ein Däne im Ruby-IRC-Channel auf, löcherte alle mit Fragen und sprach von einem Framework, das er entwickelt.

Das alles ist mittlerweile schon ein paar Jahre her. Der Kollege schreibt inzwischen selber Bücher und die Sprache, von der ich spreche, ist natürlich Ruby. Der Japaner ist der Ruby-Erfinder Yukihiro Matsumoto (»Matz«), der Amerikaner ist niemand geringerer als Dave Thomas und der Däne selbstverständlich David Heinemeier Hansson, der Erfinder von Ruby on Rails.

Wenn ich zurückblicke, muss ich immer noch ein bisschen schmunzeln. Es erschien alles so exotisch und machte neugierig. An die heutigen Auswirkungen haben damals wahrscheinlich die wenigsten gedacht.

Nach den ersten Gehversuchen in Ruby war mir klar, dass der Rückweg zu Java nicht einfach werden würde. Zurück von einer Sprache, die so mächtig in seinen Fähigkeiten und dabei stark an die menschliche Sprache angelehnt ist.

Warum ist vorher niemand auf diese Idee gekommen? Es hätte einigen von uns sehr viel Arbeit erspart.

Genauso ergeht es vielen auch mit Ruby on Rails. Einmal damit angefangen, hat man das Gefühl, dass dies die richtige Art ist, Web-Applikationen zu schreiben. Eine konsequente Trennung von Daten, Layout und Logik, außerdem viele Helfer, die zusätzlich die Entwicklung vereinfachen.

Anstatt alles in langen Konfigurationsdateien abzulegen, werden Konventionen benutzt. Sobald man diese verinnerlicht hat, will man sie nicht mehr missen und fühlt sich automatisch in den meisten Rails-Applikation heimisch.

In dem vorliegenden Buch von Tanja und Hussein bekommt der Leser einen guten Einblick in die Webentwicklung mit Ruby on Rails.

Neben den Haupt-Bereichen, wie Model, View und Controller kommen auch andere wichtige Bereiche wie Ajax, Sicherheit & Co. nicht zu kurz.

Da Rails konsequent auf Konventionen und das DRY-Prinzip (*Don't repeat yourself*) setzt, werden diese in den jeweiligen Abschnitten ausführlich erklärt und ihre Herkunft und Zweck beschrieben.

Somit wird der Leser gut auf die Arbeit mit Rails vorbereitet. Bei den jeweiligen Methoden oder Funktionen werden alle möglichen Parameter aufgelistet und kurz erklärt. Dadurch ist dieses Buch auch ein gutes Nachschlagewerk, wenn der Leser das Anfängerstadium verlassen hat.

Ich wünsche allen Leser viel Spaß bei diesem Buch, und heiße sie willkommen in der Rails-Welt!

Kaan Karaca

Herdecke

<http://www.rubyonrails.de>

Einleitung

Für wen wurde dieses Buch geschrieben?

Das Buch ist für Webentwickler geschrieben, die mit Ruby on Rails datenbankbasierte Web-Applikationen entwickeln möchten. Im Vergleich zu anderen Frameworks können Sie mit Ruby on Rails sehr schnell komplexe Web-Anwendungen entwickeln.

Die Beispielapplikationen sind systematisch von einer einfachen Applikation bis hin zu einer komplexeren Anwendung inklusive Test-Driven Development aufgebaut. Das Buch ist praxisorientiert und zum Lernen und Nachschlagen mit zahlreichen Beispielen, Tipps und Tricks hervorragend geeignet.

Es ist sehr hilfreich, wenn Sie über Datenbankkenntnisse verfügen und bereits serverseitige Websites z. B. in PHP oder Java entwickelt haben. Ruby-Kenntnisse sind nicht erforderlich. Kenntnisse in objektorientierter Programmierung wären sehr vorteilhaft. Das Buch enthält ein Kapitel zur Einführung in Ruby, in dem die Aspekte für die Arbeit mit Rails genauer betrachtet werden.

Wir selbst haben als PHP-Entwickler angefangen und entwickeln inzwischen jedes neue Projekt in Ruby on Rails. Wir sind sehr begeistert von Ruby on Rails und hoffen, dass wir Ihnen mit diesem Buch einen guten Einstieg in Ruby on Rails liefern können.

Was befindet sich in diesem Buch?

In diesem Buch werden alle Themen behandelt, die Sie für die Entwicklung von professionellen Web-Applikationen benötigen.

Im **ersten Teil**, »Grundlagen«, lernen Sie, wie Sie Ruby on Rails installieren und wie Sie eine erste kleine Applikation entwickeln. Falls Sie die Programmiersprache Ruby noch nicht kennen, finden Sie hier auch das Kapitel »Einführung in Ruby«.

Im **zweiten Teil** werden zwei Web-Applikationen entwickelt. Zunächst eine Bookmarkverwaltung, in der Schritt für Schritt gezeigt wird, was hinter den Kulissen von Ruby on Rails passiert. Die zweite Applikation wird nach dem TTD-Prinzip entwickelt.

Im **dritten Teil** gehen wir genauer auf die Struktur und die Konfiguration von Ruby on Rails und seinen einzelnen Komponenten ein. Dieser Teil dient als Vertiefung, nachdem Sie die Beispiele durchgearbeitet haben oder auch als Referenz.

Im **letzten Teil** wird gezeigt, wie Sie Ihre Applikation durch Plug-ins erweitern, vor Angriffen schützen, die Performance optimieren und die Applikation auf einem Server installieren bzw. auf einen Server übertragen.

Hinweise zu Listings

Aufgrund der Länge von einigen Befehlen, mussten im Buch einige Umbrüche gemacht werden. In Ruby- und HTML-Beispielen haben wir einfach einen Umbruch an einer zulässigen Stelle gemacht. In Kommandozeilen-Befehlen haben wir einen Backslash `\` am Ende der Zeile, die umbrochen werden musste, eingefügt.

Der Folgende Befehl kann mit dem Backslash in zwei Zeilen

```
ruby script/generate scaffold employee \  
  firstname:string
```

oder ohne Backslash in einer Zeile geschrieben werden.

```
ruby script/generate scaffold employee firstname:string
```

Wenn Sie die Befehle abtippen, können Sie sie einfach ohne Backslash in eine Zeile schreiben.

Was ist auf der CD-ROM

Auf der beiliegenden CD-ROM finden Sie neben den Code-Beispielen, auch zahlreiche Editoren und Entwicklungsumgebungen.

Die Website zum Buch

Auf der Website <http://www.railsbuch.de> finden Sie neben News auch ein Forum, in dem Sie Fragen stellen können. Da einige Links im Buch zu lang sind, haben wir in diesen Fällen die URLs als <http://railsbuch.de/urls/nr> angegeben. Wenn Sie die URLs eingeben, werden Sie entsprechend weitergeleitet.

Wie wir dieses Buch erstellt haben

Wir sind sehr froh, dass der Verlag uns die Möglichkeit gegeben hat, das Buch in \LaTeX zu schreiben. Somit konnten wir die gleichen Werkzeuge, die wir für das Programmieren mit Rails einsetzen, auch für das Schreiben des Buches verwenden. Als Texteditor haben wir Textmate verwendet und für die Versionsverwaltung haben wir Subversion eingesetzt. Dadurch konnten wir ca. 250 km voneinander entfernt, gemeinsam am Buch arbeiten.

Danke

Dieses Buch wäre ohne die Hilfe und das Verständnis einiger Personen nicht zu Stande gekommen.

Besonders danken möchten wir unseren Lebenspartnern Esmā und Jörg für ihre unendliche Geduld, Ermutigungen, und die Unterstützung, die wir während des Schreibens sehr nötig hatten.

Auch möchten wir uns bei dem luxemburgischen Reiseunternehmen Sales-Lentz bedanken, die es durch ihre innovativen Anforderungen möglich und nötig machen, dass wir zahlreiche Projekte in Ruby on Rails entwickeln.

Nicht zuletzt möchten wir uns beim Verlag, insbesondere bei Herrn Watermann, bedanken, der es uns ermöglicht hat, das Buch zu veröffentlichen. Außerdem bei Herrn Kaan Karaca für seine Unterstützung als Fachgutachter.

Hussein Morsy und **Tanja Otto**

Düsseldorf und Langsur

<http://www.railsbuch.de>

Nachdem Sie in den letzten beiden Kapiteln zwei Rails-Applikationen an praktischen Beispielen kennen gelernt haben, möchten wir Ihnen in diesem Kapitel die Struktur und die Konfiguration einer Rails-Applikation erläutern.

7 Rails-Projekte erstellen

Bevor Sie ein neues Rails-Projekt anlegen, überprüfen Sie bitte die Version Ihrer Rails-Installation. Sie sollten mindestens Rails 2.0.2 verwenden. Mit dem Befehl `rails -v` können Sie überprüfen, welche Version Sie installiert haben. Rails 2.0

```
rails -v
Rails 2.0.2
```

Sollten Sie noch eine ältere Version installiert haben, so aktualisieren Sie bitte Ihre Rails-Version (siehe Kapitel 2 ab Seite 41).

7.1 Generieren eines Rails-Projektes

Mit dem Befehl `rails pfad/projektname` wird ein neues Projekt erzeugt. Falls das Verzeichnis `projektname` nicht existiert, wird es erstellt. rails

Während der Ausführung des `rails`-Befehls wird ausgegeben, welche Verzeichnisse und Dateien erstellt werden:

```
rails pfad/name_des_projektes

create
create app/controllers
create app/helpers
...
create log/development.log
create log/test.log
```

7.1.1 Datenbank festlegen

Als Standard-Datenbanksystem wird seit Rails 2.0.2 SQLite3 verwendet. Wenn Sie ein anderes Datenbanksystem nutzen möchten, können Sie mit

Hilfe des Parameters `-d` oder `--database` angeben, welches Datenbanksystem verwendet werden soll. Diese Einstellung kann aber auch noch nachträglich jederzeit durch Anpassung der Datei `config/database.yml` geändert werden (siehe Abschnitt 7.4).

```
rails -d mysql demo1
```

```
rails --database postgresql demo2
```

7.1.2 Freeze Rails

Für die Ausführung von Rails werden normalerweise die installierten RubyGems-Pakete verwendet. Die Variable `RAILS_GEM_VERSION` der Datei `config/environment.rb` gibt an, welche Rails-Version benutzt werden soll. Wenn die Rails-Applikation z. B. mit `script/server` gestartet wird, wird überprüft, ob die entsprechende Version (z. B. 2.0.2) als RubyGems-Paket installiert ist. Ist dies nicht der Fall, so wird die Installation abgebrochen.

```
...
RAILS_GEM_VERSION = '2.0.2' ...
...
```

Listing 71 Auszug aus `config/environment.rb`

Rails updaten Sie können die Versionsnummer ändern, wenn Sie eine andere Rails-Version verwenden möchten. Angenommen, Sie haben gerade Rails 2.0.3 installiert, dann sollten Sie die Versionsnummer in `config/environment.rb` anpassen, um die neueste Version in Ihrer Rails-Applikation zu verwenden. Anschließend sollten Sie den Rake-Task `rake rails:update` ausführen.

Nicht selten tritt der Fall auf, dass Sie zwar auf Ihrem Entwicklungsrechner die richtige Version installiert haben, auf dem Server jedoch nicht. Wenn Sie selbst der Administrator des Servers sind, können Sie das schnell nachholen, sonst bitten Sie den zuständigen Administrator darum. Es gibt jedoch eine Alternative, um die Installation der spezifischen Version auf dem Server zu umgehen.

Es ist möglich, die aktuellsten RubyGems-Pakete für Rails, die auf Ihrem System installiert sind, in das Rails-Projekt zu importieren. Dieser Vorgang wird als **freeze** (»einfrieren«) bezeichnet.

Um die Rails-Version für ein Projekt »einzufrieren«, nutzen Sie beim Erzeugen des Projekts die Option `-f` oder `--freeze`. Die RubyGems-Pakete werden dann in das Verzeichnis `vendor/rails` kopiert.

```
rails -f demo3
```

Sie können die Rails-Version für ein Projekt auch im Nachhinein mit folgendem Rake-Task »einfrieren«:

```
rake rails:freeze:gems
```

Es ist auch möglich, die neueste Entwickler-Version (EdgeRails) »einzufrieren«. Siehe Abschnitt 7.12 auf Seite 319.

Mit dem Rake-Task `rake rails:unfreeze` wird das Verzeichnis `vendor/rails` wieder entfernt.

7.1.3 Rails-Befehl auf vorhandenem Projekt anwenden

Um den `rails`-Befehl auf einem bereits vorhandenen Rails-Projekt auszuführen, stehen drei Optionen zur Verfügung: Optionen

- ▶ **-p, --pretend**
Bewirkt, dass die Ausgabe des `rails`-Befehls erfolgt, ohne dass der Befehl ausgeführt wird.
- ▶ **-f, --force**
Bewirkt, dass der `rails`-Befehl ausgeführt wird und dass bereits vorhandene Dateien ohne Rückfrage überschrieben werden.
- ▶ **-s, --skip**
Bewirkt, dass der `rails`-Befehl ausgeführt wird und dass bereits vorhandene Dateien nicht überschrieben werden (ohne Rückfrage).

7.1.4 Sonstige Optionen

Folgende weitere Optionen stehen für den `rails`-Befehl zur Verfügung:

- ▶ **-r, --ruby=pfad**
Möglichkeit, den Pfad zum Ruby-Interpreter anzugeben. Ist nur interessant, wenn mehrere Ruby-Versionen installiert sind. Wird in der Regel nicht benötigt.
- ▶ **-q, --quiet**
Unterdrückt die Ausgabe während der Ausführung des `rails`-Befehls.

- ▶ **-c, - -svn**
Fügt automatisch alle generierten Ordner und Dateien dem SVN-Repository hinzu (`svn add`). Das Übertragen ins Repository (`svn commit`) muss manuell erfolgen.

7.2 Verzeichnisstruktur einer Rails-Applikation

Beim Anlegen eines Rails-Projekts wird die gesamte Projektverzeichnisstruktur angelegt. Nachfolgend werden wir diese Struktur erklären:

- ▶ **app:**
Enthält den Rails-Code. Hier finden Sie die Models, Controller, Views und ihre Helper-Dateien.
- ▶ **app/controllers:**
Enthält die Controller-Klassen, die für die Interaktion zwischen dem Benutzer und der Applikation zuständig sind. Die Controller-Dateien enden mit `_controller.rb`, wie zum Beispiel `guests_controller.rb`.
- ▶ **app/helpers:**
Enthält Hilfsklassen für den View.
- ▶ **app/models:**
Enthält die Model-Klassen, die in den meisten Fällen für den Datenbankzugriff dienen. In diesem Fall erben sie von der Klasse `ActiveRecord::Base`.
- ▶ **app/views:**
Enthält die Views bzw. die Templates. Für jeden Controller liegt ein eigenes Verzeichnis innerhalb von `app/views` vor, das genauso heißt wie der jeweilige Controller. Die Dateierdung der Views ist in der Regel `.html.erb` für HTML-Templates und `.rjs.erb` für Ajax-Skripte. Andere Endungen sind auch möglich.
- ▶ **config:**
Hier finden Sie alle Konfigurationsdateien für die Rails-Umgebung, die zum Beispiel das Routing, die Datenbankeinstellungen oder andere Einstellungen definieren.
- ▶ **db:**
In diesem Verzeichnis werden die Schema- und Migration-Dateien abgelegt. Die einzelnen Migration-Dateien finden Sie im Unterverzeichnis `migrate`, während die Datei `schema.rb`, die eine Zusammenfassung aller Migrations darstellt, sich direkt im Verzeichnis `\db` befin-

det. Mehr Informationen zu Migrations und der `schema.rb` erhalten Sie in Kapitel 10.

- ▶ **lib:**
Im Verzeichnis `lib` können Sie Code ablegen, der nicht direkt zu einem Model, einem Controller oder einem View gehört. Sie können aber auch Code, den sich Models, Controller und Views teilen, hier speichern.
- ▶ **public:**
Enthält statische Elemente, wie Bilder, Stylesheets und JavaScripts. Jede der Dateien im `public`-Verzeichnis ist direkt von außen aufrufbar. Sie können auch eigene Verzeichnisse erstellen, wie z. B. `pdfs` für PDF-Dokumente.
- ▶ **public/images:**
Enthält Bilder.
- ▶ **public/javascripts:**
Enthält JavaScripts.
- ▶ **public/stylesheets:**
Enthält Stylesheet-Dateien.
- ▶ **test:**
Enthält sämtliche Testdateien für die Rails-Applikation.
- ▶ **test/fixtures:**
Enthält Test- bzw. Beispiel-Daten für die Tests.
- ▶ **test/integration:**
Enthält die Integrationstests, mit denen die Gesamtfunktionalität oder größere Teile der Applikation getestet werden.
- ▶ **test/unit:**
Enthält u. a. die Modeltests.
- ▶ **test/functional:**
Enthält hauptsächlich Tests für die Controller-Klassen.
- ▶ **vendor:**
In diesem Verzeichnis werden Erweiterungen (Plug-ins) und gegebenenfalls auch EdgeRails installiert (siehe Abschnitt 7.12 auf Seite 319).

**Keine Components mehr**

In älteren Versionen von Rails enthielt das `components`-Verzeichnis mehrfachverwendbare Code-Teile der Views oder Controller. Ab der Version Rails 2.0 wird das `components`-Verzeichnis nicht mehr verwendet, deshalb wollen wir an dieser Stelle nicht weiter darauf eingehen.

7.3 Namenskonventionen

Magie? In diesem Abschnitt wollen wir Ihnen die automatische Namensgebung in Rails, die Rails-Newcomer mit am meisten verwirrt, erläutern. Was einen am Anfang am meisten überrascht, ist, dass Rails, wenn man ein Model `Person` aufruft, irgendwie zu wissen scheint, dass es auf eine Datenbanktabelle `people` zugreifen muss. Wirkt wie Magie, ist es aber nicht.

Wir werden in diesem Abschnitt die Default-Regeln von Rails für die Namensgebung erläutern. Alle diese Regeln können Sie in Ihren Klassen durch entsprechende Deklarationen überschreiben.

CamelCase, Unterstriche und Plural

Häufig benennen wir Variablen und Klassen mit mehreren Wörtern, ja manchmal sogar mit kurzen Phrasen. In Ruby sind dabei folgende Regeln zu beachten:

Regeln Variablennamen werden immer kleingeschrieben. Besteht der Name aus mehreren Wörtern, werden diese mit einem Unterstrich miteinander verbunden: `order_status`. Bei Klassen und Modulen verhält sich das anders. Hier wird der Name immer zusammengesrieben. Besteht er aus mehreren Wörtern, so beginnt jedes neue Wort mit einem Großbuchstaben, wie z. B. `FlightBooking`. Diese Schreibweise ist auch unter dem Begriff `CamelCase` bekannt.

Rails hat diese Konventionen von Ruby übernommen und in zweierlei Hinsicht erweitert:

Tabellennamen Rails geht davon aus, dass Tabellennamen, wie Variablennamen in Ruby, immer kleingeschrieben werden und, sollten sie aus mehreren Wörtern bestehen, dass diese durch einen Unterstrich getrennt werden. Außerdem wird vorausgesetzt, dass Tabellennamen immer im Plural stehen. Dieses führt dann zu Tabellennamen wie z. B. `employees` oder `flight_bookings`.

Dateinamen werden in Rails auch kleingeschrieben, und falls erforderlich, werden zusammengesetzte Wörter durch einen Unterstrich miteinander verbunden.

Dateinamen

Diese Annahmen nutzt Rails, um Namen automatisch zu konvertieren. Zum Beispiel könnte Ihre Applikation ein Model enthalten, das Abenteuerreisen verwaltet. Sie würden diese Klasse `AdventureTrip` nennen. Rails würde daraus automatisch ableiten, dass

- ▶ die dazugehörige Datenbanktabelle `adventure_trips` heißen würde (der Name des Models wird in Kleinbuchstaben umgewandelt, die einzelnen Wörter werden durch Unterstriche miteinander verbunden und in den Plural gesetzt).
- ▶ die Klasse in der Datei `app/models/adventure_trip.rb` definiert ist.

Für die Rails-Controller gelten zusätzliche Namenskonventionen. Angenommen, wir haben einen Controller mit dem Klassennamen `FlightsController`, dann gelten folgende Konventionen:

Controller

- ▶ Die Controller-Datei heißt `flights_controller.rb` und befindet sich im Verzeichnis `app/controllers`.
- ▶ Das zugehörige Helfermodul heißt `FlightsHelper` und wird in der Datei `flights_helper.rb` im Verzeichnis `app/helpers` definiert.
- ▶ Die dazugehörigen Templates (Views) befinden sich im Verzeichnis `app/views/flights`.
- ▶ Wenn die Datei `app/views/layouts/flights.html.erb` existiert, wird diese Layout-Datei für alle Templates des `FlightsController`s verwendet.

Es gibt noch eine weitere Besonderheit in Rails:

In »normalem« Ruby-Code müssen wir, bevor wir auf Klassen oder Module zugreifen, diese über das Keyword `require` inkludieren. Da Rails die Beziehungen zwischen Dateinamen und Klassennamen kennt, ist das normalerweise in einer Rails-Applikation nicht nötig. Stattdessen leitet Rails beim ersten Aufruf einer unbekannt Klasse oder eines unbekannt Modul aus deren bzw. dessen Namen den entsprechenden Dateinamen ab und versucht, die Datei im Hintergrund zu laden. Deshalb können Sie einfach eine Klasse aufrufen, und schon wird die Klasse in Ihre Applikation geladen.

kein require erforderlich

Das ist normalerweise so. Die einzige Ausnahme bildet Code im Verzeichnis `lib` und dessen Unterverzeichnissen. Da Rails diesen Bereich

nicht verwaltet, müssen Sie die Klassen, die Sie aus diesem Verzeichnis nutzen wollen, über `require` einbinden:

```
require "klasse_aus_lib_verzeichnis"
```

7.4 Datenbank-Konfiguration

`database.yml` Beim Generieren eines Rails-Projekts erstellt Rails die Datenbank-Konfigurationsdatei `config/database.yml` mit den Einstellungen für den Datenbankzugriff automatisch. Rails geht standardmäßig von einer SQLite3-Datenbank aus. Wenn Sie einen anderen Datenbankadapter verwenden möchten, können Sie diesen mit dem optionalen Parameter `--database` oder `-d` angeben.

► **MySQL:**

```
--database mysql
```

► **Oracle:**

```
--database oracle
```

► **PostgreSQL:**

```
--database postgresql
```

Die Option `--database` müssen Sie jedoch nicht beim Erzeugen des Projekts angeben, da Sie die Konfigurationsdatei `database.yml` jederzeit an einen anderen Datenbankadapter anpassen können.

[»] Datenbankadapter in Rails 2.0

Rails 2.0 enthält standardmäßig nur Datenbankadapter für MySQL, SQLite3 und PostgreSQL. Die Adapter der kommerziellen Datenbanken wurden in RubyGems-Pakete ausgelagert. Nach der Installation des jeweiligen RubyGems-Paketes können die Datenbankadapter wie gewohnt genutzt werden.

Die Konfigurationsdatei `database.yml` liegt im YAML-Format (siehe Anhang) vor. Nach der Generierung des Rails-Projekts ist die Datei wie folgt für SQLite3 konfiguriert:

```
# SQLite version 3.x
# gem install sqlite3-ruby (not necessary on OS X Leopard)
development:
  adapter: sqlite3
  database: db/development.sqlite3
  timeout: 5000
```

```
# Warning: The database defined as 'test' will be erased and
# re-generated from your development database when you run
# 'rake'.
# Do not set this db to the same as development or production.
test:
  adapter: sqlite3
  database: db/test.sqlite3
  timeout: 5000

production:
  adapter: sqlite3
  database: db/production.sqlite3
  timeout: 5000
```

Listing 7.2 config/database.yml

Die Einstellungen in der `database.yml` sind in drei Bereiche unterteilt:

1. **Development:**
Datenbankeinstellungen für die Entwicklungsumgebung
2. **Test:**
Datenbankeinstellungen für die Testumgebung
3. **Production:**
Datenbankeinstellungen für die Produktionsumgebung

Jede dieser Umgebungen enthält folgende Einstellungsmöglichkeiten:

1. **Adapter:**
Datenbanktyp, z. B. `mysql`, `postgresql` oder `sqlite3`
2. **Database:**
Name der Datenbank oder bei SQLite der Datenbankdatei
3. **Timeout:**
maximale Dauer für das Warten auf die Freigabe der Datenbank in Millisekunden

Wenn Sie eine MySQL-Datenbank verwenden, sind folgende Einstellungen möglich:

1. **Adapter:**
Datenbanktyp, z. B. `mysql`, `postgresql` oder `sqlite3`
2. **Encoding:**
Der zu verwendende Zeichensatz in der Datenbank

3. Database:

Name der Datenbank oder bei SQLite der Datenbankdatei

4. Username:

Benutzername für den Zugriff auf die Datenbank (nicht bei SQLite)

5. Password:

Passwort für den Zugriff auf die Datenbank (nicht bei SQLite)

6. Socket:

Socket-Datei für MySQL (wird automatisch ermittelt)

Hätten Sie bei der Generierung des Projekts nicht `rails projektname`, sondern `rails -d mysql projektname` verwendet, hätte Rails diese Einstellungen automatisch vorgenommen.

Socket oder Host? Es gibt zwei Arten, wie Rails mit einer MySQL-Datenbank kommuniziert:

1. Über eine Socket-Datei:

Schnelle Verbindung zur Datenbank, setzt allerdings voraus, dass der Pfad zu der Datei der ist, den MySQL erwartet. Diese Verbindung wird automatisch beim Erzeugen der Applikation in der `database.yml` über den Eintrag `socket:` angelegt.

2. Über das Netzwerk:

Diese Verbindung ist ein klein wenig langsamer als die Verbindung über die Socket-Datei. Es wird in der `database.yml` ein zusätzlicher Eintrag `host: 127.0.0.1` benötigt, und der `socket`-Eintrag kann entfallen. Auf Windows-basierten Systemen ist diese Konfiguration zu empfehlen. Zusätzlich kann man über den Eintrag `port` den Port angeben. Wenn Sie den Standardport 3306 für MySQL verwenden, können Sie diesen Eintrag aber weglassen.

Falls Sie SQLite3 verwenden, können Sie die Einstellungen belassen. Auch die Datenbankdatei `development.sqlite3`, in der die gesamte Datenbank gespeichert wird, hat Rails bereits im Verzeichnis `db` erstellt.

Bei anderen Datenbanken wie z. B. MySQL wird ein Datenbankserver benötigt, der installiert werden muss.

**Keine Tabulatoren in YAML-Dateien verwenden**

Achten Sie beim Bearbeiten der Konfigurationsdatei unbedingt darauf, dass Sie die Einrückung beibehalten und keine Tabulatoren, sondern nur Leerzeichen verwenden.

7.5 Umgebungseinstellungen

Im letzten Abschnitt wurde die Konfiguration der Datenbank besprochen. In diesem Abschnitt werden weitere Konfigurationsdateien vorgestellt. Eine Rails-Applikation wird in einer sogenannten Umgebung (engl. **environment**) ausgeführt. Standardmäßig sieht Rails drei Umgebungen vor:

Environment

- ▶ Entwicklungsumgebung (development)
- ▶ Produktionsumgebung (production)
- ▶ Testumgebung (test)

Für jede dieser drei Umgebungen befindet sich eine eigene Konfigurationsdatei (development.rb, production.rb und test.rb) im Verzeichnis config/environments.

Konfigurationsdateien

Die Konfiguration der Umgebungen zur Laufzeit wird von zwei Dateien bestimmt: Der config/environment.rb, die Einstellungen enthält, die unabhängig von den Umgebungen sind, und der Konfigurationsdatei für die jeweilige Umgebung.

In der environment.rb können Sie Konstanten, die in allen Umgebungen Ihrer Applikation zur Verfügung stehen sollen, definieren.

environment.rb

```
# Pfad, in dem die hochgeladenen Bilder gespeichert werden.
DIRECTORY_IMAGE_UPLOADS = "#{RAILS_ROOT}/public/image_uploads"

# Anzahl Hotels, die pro Seite dargestellt werden
HOTELS_PER_LIST = 10

# Betreff-Auswahlliste für Kontaktformular
# (contactmessages/new)
CONTACTMESSAGE_SUBJECTS = %w{Frage Reklamation Lob Kritik}
```

Subversion-Revision-Nr.

[+]

Wenn Sie die aktuelle Revision-Nr. Ihres Subversion-Repository anzeigen möchten, können Sie sie über nachfolgenden Befehl in einer Konstanten speichern und in der gesamten Applikation verwenden, wo immer Sie möchten: `APP_REVISION = IO.popen("svn info").readlines[4]`

Die Konfigurationsdateien der einzelnen Umgebungen sind standardmäßig nicht leer, sondern wie folgt konfiguriert:

7.5.1 Entwicklungsumgebung (development)

`development.rb` Die Konfigurationsdatei der Entwicklungsumgebung `development.rb` im Verzeichnis `config/environments` ist standardmäßig so konfiguriert, dass die Applikation bei jeder Anfrage neu geladen wird. Das verzögert zwar die Reaktionszeit, hat aber den Vorteil, dass Sie nicht nach jeder Änderung am Code den Server neu starten müssen. Außerdem werden die Fehlermeldungen protokolliert, wenn Sie eine Methode auf einem `nil`-Objekt aufrufen. Es werden ausführliche Fehlermeldungen mit Codeauszügen angezeigt, und das Caching ist deaktiviert. Innerhalb der Entwicklungsumgebung wird standardmäßig ignoriert, wenn E-Mails aus der Applikation heraus nicht versendet werden können:

```
# Settings specified here will take precedence over those in
# config/environment.rb

# In the development environment your application's code is
# reloaded on every request.
# This slows down response time but is perfect for development
# since you don't have to restart the webserver when you make
# code changes.
config.cache_classes = false

# Log error messages when you accidentally call methods
# on nil.
config.whiny_nils = true

# Show full error reports and disable caching
config.action_controller.consider_all_requests_local = true
config.action_view.debug_rjs                         = true
config.action_controller.perform_caching             = false
config.action_view.cache_template_extensions         = false

# Don't care if the mailer can't send
config.action_mailer.raise_delivery_errors = false
```

Wenn Sie die Rails-Applikation lokal mit `ruby script/server` oder `ruby script/console` starten, werden die Einstellungen aus der `environment.rb` im Verzeichnis `config` und der `development.rb` im Verzeichnis `config/environments` geladen.

7.5.2 Produktionsumgebung (production)

`production.rb` Standardmäßig ist die Konfigurationsdatei der Produktionsumgebung `config/environments/production.rb` so konfiguriert, dass die Applikation

nicht bei jeder Anfrage neu geladen wird. Das heißt, die Applikation muss bei jeder Code-Änderung neu gestartet werden. Statt ausführlicher Fehlermeldungen werden nur allgemeine Fehlermeldungen ohne Codeauszüge ausgegeben, und das Caching ist aktiviert.

Änderungen an View-Dateien werden nicht mehr erkannt

[«]

Neu in Rails 2.0.2 ist, dass in der Produktionsumgebung (einstellbar durch `monoconfig.action_view.cache_template_loading`) nicht bei jedem Zugriff auf die View-Dateien überprüft wird, ob diese sich geändert haben. Durch den verringerten Festplattenzugriff steigt die Performance der Applikation. Um Änderungen an View-Dateien wirksam zu machen, muss daher die Rails-Applikation neu gestartet werden.

```
# Settings specified here will take precedence over those in
# config/environment.rb

# The production environment is meant for finished, "live"
# apps. Code is not reloaded between requests
config.cache_classes = true

# Use a different logger for distributed setups
# config.logger = SyslogLogger.new

# Full error reports are disabled and caching is turned on
config.action_controller.consider_all_requests_local = false
config.action_controller.perform_caching           = true
config.action_view.cache_template_loading           = true

# Enable serving of images, stylesheets, and javascripts from
# an asset server
# config.action_controller.asset_host =
# "http://assets.example.com"

# Disable delivery errors, bad email addresses will be ignored
# config.action_mailer.raise_delivery_errors = false
```

Wenn Sie die Applikation lokal mit `ruby script/server -e production` oder `ruby script/console production` starten, werden die Einstellungen aus der `config/environment.rb` und der `production.rb` im Verzeichnis `config/environments` geladen.

7.5.3 Testumgebung (test)

`test.rb` Die Testumgebung `config/environments/test.rb` dient ausschließlich zur Ausführen der Testklassen (siehe Kapitel 6). Sie werden sie nie in einem anderen Kontext benötigen. Während der Ausführung der Tests wird die Datenbank immer wieder verändert und zurückgesetzt.

Standardmäßig ist die Testumgebung so konfiguriert, dass die Applikation nicht bei jeder Anfrage neu geladen wird. Die Fehlermeldungen, wenn eine Methode auf ein `nil`-Objekt angewendet wird, werden protokolliert. Es werden ausführliche Fehlermeldungen mit Codeauszügen ausgegeben, und das Caching ist deaktiviert. Darüber hinaus ist der Schutz vor CSRF-Angriffen (siehe Kapitel 17 ab Seite 647) deaktiviert. E-Mails werden nicht real verschickt, sondern in ein Array ausgegeben:

```
# Settings specified here will take precedence over those in
# config/environment.rb

# The test environment is used exclusively to run your
# application's test suite.
# You never need to work with it otherwise. Remember that
# your test database is "scratch space" for the test suite
# and is wiped and recreated between test runs.
# Don't rely on the data there!
config.cache_classes = true

# Log error messages when you accidentally call methods on
# nil.
config.whiny_nils = true

# Show full error reports and disable caching
config.action_controller.consider_all_requests_local = true
config.action_controller.perform_caching           = false

# Disable request forgery protection in test environment
config.action_controller.allow_forgery_protection  = false

# Tell ActionMailer not to deliver emails to the real world.
# The :test delivery method accumulates sent emails in the
# ActionMailer::Base.deliveries array.
config.action_mailer.delivery_method = :test
```

Zum Ausführen der Testklassen wird die Applikation mit den Einstellungen aus der `config/environment.rb` und der `test.rb` im Verzeichnis `config/environments` geladen.

7.5.4 Eigene Umgebungen anlegen

Die drei gezeigten Umgebungen `development`, `production` und `test` sind die Umgebungen, die standardmäßig von Rails erzeugt werden. Aber selbstverständlich können Sie zusätzlich eigene Umgebungen definieren.

Eine beliebte zusätzliche Umgebung ist z. B. die `staging`-Umgebung (zu Deutsch: Arbeitsbühne), die dazu dient, die Veröffentlichung der Applikation zu testen. `staging`

Um eine eigene Umgebung anzulegen, gehen Sie wie folgt vor:

1. Neue Konfigurationsdatei anlegen

Im Verzeichnis `config/environments` legen Sie eine neue Konfigurationsdatei an, die so heißt wie Ihre Umgebung, z. B. `staging.rb`. In dieser Datei nehmen Sie Ihre Konfigurationseinstellungen vor.

2. Eintrag in `database.yml` hinzufügen

In der Datenbankkonfigurationsdatei `config/database.yml` definieren Sie einen zusätzlichen Bereich für die neue Umgebung, z. B.:

```
staging:
  adapter: mysql
  encoding: utf8
  database: demo1_staging
  username: root
  password:
  socket: /opt/local/var/run/mysql5/mysqld.sock
```

3. Datenbank generieren

Die in der `database.yml` neu definierte Datenbank `demo1_staging` legen Sie mit dem Befehl `rake db:create RAILS_ENV=staging` an.

Die Applikation starten Sie lokal mit den Umgebungseinstellungen der `staging`-Umgebung mit den Befehlen `ruby script/server -e staging`.

Die Konsole in der `staging`-Umgebung können Sie mit dem Befehl `ruby script/console staging` starten.

7.5.5 Clean up your environment

Vor Rails 2.0 wurde auch jede Art von einmaligen Konfigurationen innerhalb der `environment.rb` definiert. Jetzt können Sie diese Elemente in eigene Dateien im Verzeichnis `config/initializers` ablegen. Alle Dateien in diesem Verzeichnis werden automatisch geladen. Dadurch soll `config/initializers`

sichergestellt werden, dass nur die Standardeinstellungen innerhalb der `environment.rb` definiert werden.

Sie können z. B. eine Datei `meine_einstellungen` im Verzeichnis `config/initializers` anlegen, in der Sie u. a. Konstanten speichern (z. B. `ADMIN_EMAIL="admin@example.com"`). Diese Datei wird beim Start automatisch ausgeführt.

[+]

Default-Umgebung ändern

Standardmäßig wird beim Aufruf von `ruby script/server` oder `ruby script/console` die Entwicklungsumgebung (`development`) geladen.

Auf UNIX-basierten Systemen können Sie eine Umgebungsvariable setzen, um das zu ändern:

```
export RAILS_ENV = production
```

Jetzt wird beim Aufruf von `script/server` oder `script/console` die Produktionsumgebung (`production`) geladen.

7.6 Generatoren

Generatoren helfen dem Entwickler bei der Erstellung von Applikationen, indem sie Code generieren. Sie bieten folgende Vorteile:

- ▶ Hilft Anfängern und Profis.
- ▶ Verkürzt die Entwicklungszeit.
- ▶ Tests werden mit erstellt.
- ▶ Erstellung eigener Generatoren ist möglich.

7.6.1 Verwendung

`script/generate` Mit dem Befehl `ruby script/generate` erhalten Sie eine Übersicht aller Generatoren.

Um eine Beschreibung zu einem Generator zu sehen, geben Sie `ruby script/generate` gefolgt vom Namen des Generators an. Um z. B. die Beschreibung zum `controller`-Generator aufzurufen geben Sie `ruby script/generate controller` ein:

```
ruby script/generate controller
```

```
Usage: script/generate controller ControllerName [options]
```

Description:

```
  Stubs out a new controller and its views. Pass the
  controller name, either CamelCased or under_scored,
  and a list of views as arguments.
```

```
...
```

7.6.2 Übersicht aller Generatoren

Folgende Generatoren werden mit Rails geliefert:

► Controller:

Controller dienen der Interaktion zwischen dem Benutzer und der Rails-Applikation. Der Generator generiert eine Controller-Klasse im Verzeichnis `app/controllers`, den dazugehörigen funktionalen Test im Verzeichnis `test/functional`, eine Hilfsklasse für die Views im Verzeichnis `app/helpers` und die Views im Verzeichnis `app/views`, wenn diese angegeben wurden. Der erste Parameter gibt den Namen des Controllers an. Danach folgen optional die Views.

```
ruby script/generate controller Guest index show edit
```

► Integration_test:

Hiermit wird ein Integrations-Test im Verzeichnis `test/integration` generiert. Ein Integrations-Test dient zur Prüfung der Funktionsfähigkeit der Applikation.

```
ruby script/generate integration\_test Booking
```

► Mailer:

Setzen Sie diesen Generator ein, wenn Sie aus Ihrer Rails-Applikation heraus E-Mails versenden möchten (siehe Kapitel 11 ab Seite 523). Es werden neben der Mailer-Klasse im Verzeichnis `app/models` Tests im Verzeichnis `test/unit` mit Testdaten (Fixtures) im Verzeichnis `test/fixtures` und optional auch die Views generiert.

```
ruby script/generate mailer ContactmessageMailer confirm
```

► Migration:

Dieser Generator generiert eine Datenbank-Migration-Datei im Verzeichnis `db/migrate`. Verwenden Sie diesen Generator, um Änderungen an Ihrer Datenbankstruktur vorzunehmen, wie z. B. das Hinzufügen eines Feldes zu einer Datenbank-Tabelle. Migrations verwenden Sie auch, um die Datenbank-Tabellen zu erstellen. Für diesen Zweck

ist jedoch der Model-Generator besser geeignet.

```
ruby script/generate migration AddPriceToReservation
```

► **Model:**

Um in Ihrer Rails-Applikation auf eine Datenbanktabelle zugreifen zu können, benötigen Sie eine Model-Klasse. Dieser Generator erzeugt neben der Model-Klasse im Verzeichnis `app/models` mit den zugehörigen Tests in `test/unit` und Testdaten im Verzeichnis `test/fixtures` auch eine Datenbank-Migration-Datei für die Erstellung der Datenbank-Tabelle. Geben Sie den Namen des Models an. Optional können Sie auch schon die Felder mit den dazugehörigen Datentypen angeben:

```
ruby script/generate model Guest fname:string lname:string
```

► **Observer:**

Es wird eine Observer-Klasse im Verzeichnis `app/models` und ein zugehöriger Test im Verzeichnis `test/unit` angelegt:

```
ruby script/generate observer Reservation
```

► **Plugin:**

Setzen Sie diesen Generator ein, wenn Sie eine eigene Erweiterung erstellen möchten. Der Generator generiert ein neues Plug-in im Verzeichnis `vendor/plugins` mit einer `init.rb` und einer `README`. Zusätzlich werden die Verzeichnisse `lib`, `task` und `test` angelegt. Als Parameter erwartet der Generator den Namen des Plug-ins:

```
ruby script/generate plugin MeinPlugin
```

► **Resource:**

Diesen Generator können Sie nutzen, um eine Ressource zu erzeugen. Es werden ein leeres Model, ein leerer Controller sowie die Unit- und Functional-Testklassen und die Fixtures generiert. Der dazugehörige `map.resources`-Eintrag in der Datei `config/routes.rb` wird auch vorgenommen. Der Generator `resource` generiert keine Methoden im Controller und auch nicht die dazugehörigen Views. Das übernimmt der `scaffold`-Generator. Als Parameter erwartet der `resource`-Generator den Namen des Models im Singular und eine optionale Liste von `Spaltenname:sql_type`-Paaren. Werden diese angegeben, werden die entsprechenden Felder in der Migration angelegt. Die Felder `created_at` und `updated_at` werden automatisch angelegt.

```
ruby script/generate resource person name:string
```

► **Scaffold:**

Dieser Generator erzeugt alle erforderlichen Dateien mit allen erforderlichen Inhalten, um sofort eine Ressource nutzen zu können.

Es werden das Model, die Migration-Datei, der Controller mit den CRUD-Actions, die dazugehörigen Views und die fertigen Testklassen generiert. Der erforderliche `map.resources`-Eintrag in der Datei `config/routes.rb` wird auch automatisch vorgenommen. Als erster Parameter wird der Name des Models erwartet. Optional können Sie eine Liste von `Spaltenname:sql_type`-Paaren übergeben. Werden diese angegeben, werden die entsprechenden Felder in der Migration angelegt. Die Felder `created_at` und `updated_at` werden automatisch angelegt.

```
ruby script/generate scaffold post message:text
```

► **Session_migration:**

Mit diesem Generator erstellen Sie eine Migration, um die Tabelle `sessions` für das Session-Management anzulegen. Der Generator erwartet keine weiteren Parameter:

```
ruby script/generate session_migration
```

Weitere Generatoren stehen als Gem-Pakete zur Verfügung (`gem search -r generator`).

Rails 2.0

In Rails 2.0 wurde der Generator `scaffold_resource` aus Rail 1.2 in `scaffold` umbenannt. Der alte `scaffold`-Generator fällt ersatzlos weg.

[«]

7.6.3 Rückgängig machen

Die generierten Skripte können mit dem Befehl `ruby script/destroy` rückgängig gemacht werden. Verwenden Sie dabei die gleichen Parameter wie beim Generieren mit `ruby script/generate`.

```
ruby script/destroy controller Guest index show edit
```

```
rm app/views/guest/edit.rhtml
rm app/views/guest/show.rhtml
rm app/views/guest/index.rhtml
rm app/helpers/guest_helper.rb
rm test/functional/guest_controller_test.rb
rm app/controllers/guest_controller.rb
rmdir test/functional
notempty test
rmdir app/views/guest
notempty app/views
notempty app
notempty app/helpers
```

```
notempty app
notempty app/controllers
notempty app
```

7.7 Rails-Konsole

`ruby script/console` Mit dem Befehl `ruby script/console` starten Sie die Rails-Konsole. Innerhalb der Rails-Konsole stehen Ihnen alle Klassen Ihrer Applikation zur Verfügung. Das heißt, Sie können hier z. B. auf alle Models Ihrer Applikation zugreifen, um nach ActiveRecord-Objekten zu suchen, sie zu testen oder sie sogar zu verändern.

In PHP würde man dafür direkt auf die Datenbank zugreifen. In Rails macht man das nicht, sondern arbeitet über die Konsole mit den Objekten. Die Konsole wird häufig eingesetzt, sogar auf dem Produktionsserver.

Beim Aufruf von `ruby script/console` wird, wie bereits im Abschnitt 7.5 erwähnt, standardmäßig die Entwicklungsumgebung geladen. Wenn Sie eine andere Umgebung, zum Beispiel die Produktionsumgebung, laden wollen, übergeben Sie dem Befehl den Namen der Umgebung:

```
ruby script/console production
```

Wenn Sie die Konsole nur zum Testen von ActiveRecord-Objekten nutzen wollen, also sicherstellen wollen, dass Sie keine Werte verändern, können Sie die Konsole mit dem Parameter `-s` oder `--sandbox` aufrufen. Das bewirkt, dass alle Änderungen beim Verlassen der Konsole wieder rückgängig gemacht werden.

Um zum Beispiel die Daten des ersten Mitarbeiters aus unserer Beispiel-Applikation `employees` aus Kapitel 3 ab Seite 59 in der Konsole auszugeben, starten wir zunächst aus dem Projektverzeichnis heraus die Konsole und suchen den ersten Datensatz. Als Rückgabewert wird der Datensatz ausgegeben:

```
ruby script/console

Loading development environment (Rails 2.0.2)
>> employee = Employee.find(1)
=> #<Employee id: 1, firstname: "Lee", lastname: "Adama",
birthday: "2007-04-16", email: "lee.adama@railsair.com",
comment: "Wird auch 'Apollo' genannt",
created_at: "2007-12-23 09:50:31",
updated_at: "2007-12-23 09:50:31", freelancer: nil,
country: nil, title: nil, department_id: nil>
```

Mit Hilfe der Methode `y` können wir die Ausgabe im YAML-Format formatieren: Methode `y`

```
>> y employee
--- !ruby/object:Employee
attributes:
  updated_at: 2007-12-23 09:50:31
  title:
  country:
  id: "1"
  firstname: Lee
  lastname: Adama
  birthday: "2007-04-16"
  freelancer:
  department_id:
  comment: Wird auch "Apollo" genannt
  email: lee.adama@railsair.com
  created_at: 2007-12-23 09:50:31
attributes_cache:
```

Bei Änderungen am Code Konsole neu starten

[!]

Wenn Sie in der Konsole arbeiten, während Sie entwickeln, müssen Sie darauf achten, dass Sie, nachdem Sie Änderungen am Code vorgenommen haben, entweder die Konsole beenden (`exit`) und neu starten oder mit dem Befehl `reload!` die Applikation neu laden.

7.8 Lokaler Server

Rails bringt standardmäßig WEBrick als lokalen Server mit, in dem Sie die Applikation starten können. Diesen lokalen Server starten Sie mit dem Befehl `ruby script/server` ruby script/server

```
ruby script/server
```

```
=> Booting Mongrel (use 'script/server webrick' to force
  WEBrick)
=> Rails application starting on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
** Starting Mongrel listening at 0.0.0.0:3000
** Starting Rails with development environment...
** Rails loaded.
...
```

Mongrel Haben Sie auf Ihrem System den Webserver Mongrel installiert (Mac OS X und InstantRails bringen z. B. Mongrel mit), wird dieser gestartet.

Standardmäßig wird beim Aufruf von `ruby script/server` der Port 3000 verwendet. Das heißt, Sie können Ihre Applikation im Browser über die Adresse `http://localhost:3000` erreichen.

Mit Hilfe des Parameters `-p` oder `--port` können Sie auch einen anderen Port angeben. Das ist zum Beispiel dann sinnvoll, wenn Sie mehrere Applikationen gleichzeitig starten möchten.

Logdatei Innerhalb des Konsole-Fensters, in dem Sie den lokalen Server gestartet haben, wird die Logdatei live angezeigt. Das heißt, Sie können genau verfolgen, was gerade passiert. Sie können sehen, welche Seite im Browser aufgerufen wurde und ob es eventuell zu Fehlern kam. Zur Fehleranalyse ist diese Live-Ausgabe der Logdatei sehr gut geeignet.

Durch Aufruf des Befehls `ruby script/server` wird standardmäßig die Entwicklungsumgebung geladen (siehe Abschnitt 7.5 auf Seite 299). Wenn Sie eine andere Umgebung laden möchten, geben Sie über den Parameter `-e` den Namen der gewünschten Umgebung an:

```
ruby script/server -e production
```

Beendet wird der lokale Server durch die Tastenkombination **(Strg)+⌘**.

7.9 Logging

Beim Generieren einer Rails-Applikation wird für die drei Standardumgebungen im Verzeichnis `log` eine eigene Logdatei angelegt:

- ▶ `log/development.log`
- ▶ `log/production.log`
- ▶ `log/test.log`

Es wird immer in die zu der Umgebung gehörende Logdatei geschrieben, in der der Server gestartet wurde.

Rails protokolliert in den Logdateien sehr viele Informationen, wie z. B., welche URL aufgerufen wurde, welcher Controller und welche Action ausgeführt wurden oder welche SQL-Befehle ausgeführt wurden.

Eigene Einträge Mit Hilfe des Objekts `logger` können Sie z. B. aus einem Controller oder einem Model heraus auch einen Eintrag in der Logdatei vornehmen. Dazu stehen drei Methoden zur Verfügung:

- ▶ **logger.info(text)**
Um allgemeine Informationen zu protokollieren.
- ▶ **logger.warn(text)**
Um Warnungen, dass z. B. etwas nicht funktioniert wie erwartet, zu protokollieren.
- ▶ **logger.debug(text)**
Um Variablen auszugeben, die der Fehlersuche dienen können. Standardmäßig wird dieser Befehl in der Produktionsumgebung ignoriert, d. h., es wird kein Eintrag in der Logdatei vorgenommen.

Ein Aufruf könnte beispielsweise wie folgt lauten:

```
logger.debug("User-id = #{user.id}")
```

7.10 Debugging

Ab Rails 2.0 ist wieder der Ruby-Debugger in Rails enthalten. Ein Debugger (wörtl. »Entwanzer«) hilft beim Analysieren von Fehlern (bzw. Bugs), indem man in der Applikation einen Haltepunkt (engl. **Breakpoint**) setzt. Trifft die Applikation auf diesen Breakpoint, wird die Applikation angehalten, und man kann in einer Konsole u. a. die Werte der Variablen überprüfen. Mit dem sogenannten **Continue**-Befehl wird die Ausführung der Applikation wieder an der gleichen Stelle fortgesetzt, bis die Applikation gegebenenfalls wieder auf einen Breakpoint stößt.

7.10.1 Installation von ruby-debug

Um ihn einsetzen zu können, müssen Sie das Gem-Paket `ruby-debug` installieren:

```
sudo gem install ruby-debug
```

Auf Windows-basierten Systemen entfällt das vorangestellte `sudo`.

Mac OS X-User

Mac OS X-User müssen vor der Installation des Gem-Paketes `ruby-debug` die XCode-Tools von der Installations-CD installieren, da ein GCC-Compiler benötigt wird.

[«]

7.10.2 Breakpoint setzen

Um den Debugger zu nutzen, setzen Sie im Model oder Controller an der Stelle, an der Sie den Debugger nutzen wollen, den Befehl `debugger` ein. Im folgenden Beispiel wird der Breakpoint-Befehl in der `create`-Methode des People-Controllers gesetzt.

```
def create
  @person = Person.new(params[:person])
  debugger
  ...
end
```

Listing 7.3 `app/controllers/people_controller.rb`

In diesem Beispiel haben wir den Breakpoint direkt nach dem Erzeugen des Person-Objekts eingefügt, um zu analysieren, welche Attribute das Objekt erhalten hat.

7.10.3 Server mit Debugger starten

Starten Sie den Debugger gemeinsam mit dem lokalen Server mit dem Befehl `ruby script/server --debugger` oder `ruby script/server -u`.

Rufen Sie dann im Browser die Stelle Ihrer Applikation auf, an der Sie den Debugger gesetzt haben. Rufen Sie in unserem Beispiel die URL `http://localhost:3000/people/new` auf, geben Sie ein Beispiel ein, und klicken Sie auf die Schaltfläche »Create«.

Stehender
Ladebalken

Die Ausführung der betreffenden Action wird beim Erreichen des Debugger-Befehls gestoppt. Im Safari-Browser kann man ganz gut erkennen, dass der Ladebalken in der URL stehen bleibt.

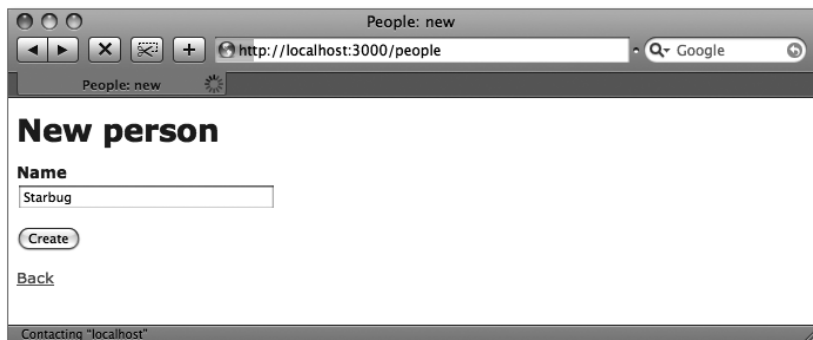


Abbildung 7.1 New-Seite im Safari-Browser

7.10.4 Debugger-Konsole

In dem Konsolen-Fenster, in dem Sie den Debugger gestartet haben, haben Sie jetzt vollen Zugriff auf die zur Zeit zur Verfügung stehenden Objekte.

Sie können die Objekte ausgeben (`y @objekt`) oder verändern (`@objekt.name = "neuer_name"`).

In unserem Beispiel können wir z. B. die Attribute des Objekts `@person` anzeigen.

```
(rdb:13) y @person
--- !ruby/object:Person
attributes:
  name: Starbug
  updated_at:
  created_at:
attributes_cache:

new_record: true
nil
```

Listing 7.4 Debugger-Konsole

Praktisch ist es auch, im Controller die übergebenen Parameter `params` oder die Umgebungsvariablen `env` auszugeben.

```
(rdb:15) y params
--- !map:HashWithIndifferentAccess
commit: Create
authenticity_token: f8df001e3459d25917a3bc2c5a68362e372e8a3d
action: create
controller: people
person: !map:HashWithIndifferentAccess
  name: Starbug
nil
```

Listing 7.5 Ausgabe von params und env in der Debugger-Konsole

7.10.5 Ausführung fortsetzen

Durch Eingabe des Befehls `continue` in der Konsole wird der Code weiter ausgeführt. Der `continue`-Befehl ist ein Befehl des Ruby-Debuggers. Mit `help` können Sie alle verfügbaren Befehle auflisten lassen. `continue`

Mehr Informationen zum Debugger und zu seinen Möglichkeiten finden Sie auf der Website <http://www.datanoise.com/ruby-debug/>.

7.11 Rake

Rake ist ein Tool, das in der Softwareentwicklung eingesetzt wird. Es ist in Ruby geschrieben, und die Rake-Files, die äquivalent zu den Make-Files von `make` sind, benutzen die Ruby-Syntax. Rake wurde von Jim Weirich entwickelt.

Tasks Rake benutzt anonyme Funktionsblöcke von Ruby, um verschiedene Tasks zu definieren, und erlaubt den Gebrauch der Ruby-Syntax. Es gibt eine Bibliothek für die allgemeinen Tasks wie z. B. die Dateibearbeitung und eine Bibliothek, um bereits kompilierte Dateien zu entfernen (`clean-task`). Wie `make`, kann `rake` auch Tasks ausführen, die auf Mustern basieren.

Eine Übersicht aller Rake-Tasks erhalten Sie mit `rake -tasks` oder in der kürzeren Schreibweise mit `rake -T`. Durch die Installation von Erweiterungen (Plug-ins) kann die Liste auch länger sein. Die Rake-Tasks bzgl. Datenbanken (`rake db:*`) werden ausführlich im Abschnitt 10.3 auf Seite 440 behandelt.

Folgende Rake-Tasks sind standardmäßig verfügbar:

- ▶ **rake db:abort_if_pending_migrations**
Zeigt an, welche Migrations noch nicht mit `rake db:migrate` ausgeführt wurden.
- ▶ **rake db:charset**
Gibt den verwendeten Zeichensatz der Datenbank aus.
- ▶ **rake db:collation**
Gibt die verwendete Kollation (Sortierung) der Datenbank aus.
- ▶ **rake db:create**
Erzeugt die Datenbank, die in `config/database.yml` für die aktuelle Umgebung definiert ist.
- ▶ **rake db:create:all**
Erzeugt lokal alle Datenbanken, die in `config/database.yml` definiert sind.
- ▶ **rake db:drop**
Löscht die Datenbank der aktuellen Umgebung.

- ▶ **rake db:drop:all**
Löscht alle lokalen Datenbanken, die in `config/database.yml` definiert sind.
- ▶ **rake db:fixtures:identify**
Sucht nach einem Fixture mit einem Label oder einer ID.
- ▶ **rake db:fixtures:load**
Lädt die Fixtures in die aktuelle Datenbank der Entwicklungsumgebung. Wird der optionale Parameter `FIXTURES=x,y` gesetzt, werden explizit nur diese Fixtures geladen.
- ▶ **rake db:migrate**
Führt die Migration-Skripte im Ordner `db/migrate` aus und ändert die Datenbankstruktur entsprechend. Durch Angabe einer bestimmten Migration über den optionalen Parameter `VERSION=x` kann man die Datenbank auf den Stand dieser Version bringen.
- ▶ **rake db:migrate:redo**
Es wird die letzte Migration rückgängig gemacht (Rollback) und anschließend wieder ausgeführt. Die Anzahl der Versionen, um die zurückgegangen werden soll, kann über `STEP=n` angegeben werden.
- ▶ **rake db:migrate:reset**
Es wird die Datenbank gelöscht, dann die Datenbank erstellt und alle Migrations ausgeführt.
- ▶ **rake db:reset**
Löscht die Datenbank der aktuellen Umgebung und erzeugt sie wie in `db/schema.rb` definiert neu.
- ▶ **rake db:rollback**
Es wird die letzte Migration rückgängig gemacht. Die Anzahl der Versionen, um die zurückgegangen werden soll, kann über `STEP=n` angegeben werden.
- ▶ **rake db:schema:dump**
Erstellt eine `schema.rb`-Datei im Ordner `db`, die in jede Datenbank geladen werden kann, die von ActiveRecord unterstützt wird.
- ▶ **rake db:schema:load**
Lädt eine `schema.rb`-Datei in die Datenbank.
- ▶ **rake db:sessions:clear**
Löscht alle Sessions.

- ▶ **rake db:sessions:create**
Erstellt eine Migration-Datei für die Sessionverwaltung mit `CGI::Session::ActiveRecordStore`.
- ▶ **rake db:structure:dump**
Gibt die Datenbankstruktur in eine SQL-Datei aus.
- ▶ **rake db:test:clone**
Erstellt die Datenbank der Testumgebung anhand des aktuellen Datenbankschemas der Entwicklungsumgebung neu.
- ▶ **rake db:test:clone_structure**
Erstellt die Datenbank der Testumgebung anhand der aktuellen Struktur der Entwicklungsumgebung neu.
- ▶ **rake db:test:prepare**
Erstellt die Testdatenbank und lädt das Schema.
- ▶ **rake db:test:purge**
Leert die Testdatenbank.
- ▶ **rake db:version**
Gibt die aktuelle Versionsnummer des Schemas zurück.
- ▶ **rake doc:app**
Generiert die Dokumentation der Applikation im HTML-Format.
- ▶ **rake doc:clobber_app**
Löscht die Dokumentation der Applikation.
- ▶ **rake doc:clobber_plugins**
Löscht die Dokumentation der Plug-ins.
- ▶ **rake doc:clobber_rails**
Löscht die Dokumentation der Applikation
- ▶ **rake doc:plugins**
Generiert die Dokumentation aller Plug-ins im HTML-Format.
- ▶ **rake doc:rails**
Generiert die Rails-Dokumentation im HTML-Format.
- ▶ **rake doc:reapp**
Erzeugt eine neue Dokumentation der Applikation und überschreibt die alte.
- ▶ **rake doc:reraails**
Generiert die RDOC-Dateien neu.

- ▶ **rake log:clear**
Leert alle Log-Files im Ordner `/log`.
- ▶ **rake notes**
Gibt die Anzahl aller Anmerkungen zurück.
- ▶ **rake notes:fixme**
Gibt die Anzahl aller FIXME-Anmerkungen zurück.
- ▶ **rake notes:optimize**
Gibt die Anzahl aller OPTIMIZE-Anmerkungen zurück.
- ▶ **rake notes:todo**
Gibt die Anzahl aller TODO-Anmerkungen zurück.
- ▶ **rake rails:freeze:edge**
Es wird die neueste Entwickler-Version von Rails (Edge) in das Verzeichnis `vendor/rails` installiert. Mit der Option `REVISION` können Sie die Revision wählen (z. B. `REVISION=8679`), und mit der Option `TAG` (z. B. `TAG=rel_2-0-2`) können Sie den Namen der Rails-Version angeben, die Sie aus dem SVN-Repository von Rails laden möchten.
- ▶ **rake rails:freeze:gems**
Es werden die aktuellen RubyGems-Pakete für Rails, die auf Ihrem System installiert sind, in das Verzeichnis `vendor/rails` kopiert.
- ▶ **rake rails:unfreeze**
Löscht das Verzeichnis `vendor/rails`.
- ▶ **rake rails:update**
Führt ein Rails-Update für `config`, `script` und `public/javascripts` aus.
- ▶ **rake rails:update:configs**
Führt ein Update für `config/boot.rb` der aktuellen Rails-Installation durch.
- ▶ **rake rails:update:javascripts**
Führt ein Update der Javascripts der aktuellen Rails-Installation durch.
- ▶ **rake rails:update:scripts**
Fügt neue Skripte im Ordner `script/` der Applikation hinzu.
- ▶ **rake routes**
Gibt alle Routing-Einträge alphabetisch sortiert aus.

- ▶ **rake stats**
Gibt Statistiken der Applikation aus, wie z. B. die Gesamtanzahl der Zeilen an Programmiercode.
- ▶ **rake test**
Führt alle Unit- und Functional-Tests aus.
- ▶ **rake test:functionals**
Führt die Functional-Tests in `test/functional` aus.
- ▶ **rake test:integration**
Führt die Integration-Tests in `test/integration` aus.
- ▶ **rake test:plugins**
Führt die Plug-in-Tests oder den Test für ein bestimmtes Plug-in (PLUGIN=name) in `vendor/plugins/**/test` aus.
- ▶ **rake test:recent**
Testet die letzten Änderungen.
- ▶ **rake test:uncommitted**
Testet die letzten Änderungen seit dem letzten Check-in (ist nur möglich, wenn SVN genutzt wird).
- ▶ **rake test:units**
Führt die Unit-Tests in `test/unit` aus.
- ▶ **rake tmp:cache:clear**
Löscht alle Dateien und Ordner im Verzeichnis `tmp/cache`.
- ▶ **rake tmp:clear**
Löscht alle temporären Dateien im Verzeichnis `tmp`.
- ▶ **rake tmp:create**
Erzeugt die Verzeichnisse `sessions`, `cache` und `sockets` im Ordner `tmp`.
- ▶ **rake tmp:pids:clear**
Löscht alle Dateien im Verzeichnis `tmp/pids`.
- ▶ **rake tmp:sessions:clear**
Löscht alle Dateien im Verzeichnis `tmp/sessions`.
- ▶ **rake tmp:sockets:clear**
Löscht alle Dateien im Verzeichnis `tmp/sockets`.

Rake-Tasks werden aus einem Projektverzeichnis heraus aufgerufen. Zum Beispiel:

```
rake db:migrate
```

7.12 EdgeRails

In Open Source-Projekten gibt den sogenannten »trunk« – den neuesten Sourcecode. Im Gegensatz zu den »stabilen« Versionen, die frei von kritischen Bugs und für den allgemeinen Gebrauch geeignet sind, ist der Trunk die aktuelle Spitze des Source-Eisbergs. Der Trunk verändert sich dauernd, weil er ständig weiterentwickelt wird. trunk

Gute Open Source-Projekte, wie Rails, versuchen immer, eine relativ stabile Version im Trunk zur Verfügung zu stellen. Da nur die Core-Entwickler Code in den Trunk einchecken dürfen, kann man sich auch ziemlich sicher darauf verlassen, dass das gelingt. Trotzdem bleibt es das eigene Risiko, den Code im Trunk zu benutzen. Es gibt keine Garantie, dass der Trunk-Code wirklich immer fehlerfrei funktioniert.

In Rails heißt der Trunk-Code »EdgeRails«, und wenn Sie Ihre Rails-Applikation von Gem auf EdgeRails umstellen wollen, führen Sie einfach folgenden Befehl im Root-Verzeichnis Ihrer Applikation aus:

```
rake rails:freeze:edge
```

Die Dateien werden aus dem SVN-Repository mit der URL <http://svn.rubyonrails.org/rails/trunk> geladen und in das Verzeichnis `vendor/rails` exportiert.

Um die Konfigurationsdateien und JavaScript-Bibliotheken zu aktualisieren, sollten Sie anschließend noch den Rake-Task `rake rails:update` ausführen.

7.13 Ein Rails-Projekt in Subversion überführen

Versionsverwaltung wird in der Softwareentwicklung nicht nur zur Versionierung eingesetzt, sondern auch, um den gemeinsamen Zugriff mehrerer Programmierer auf den Quelltext der einzelnen Dateien zu steuern. Das heißt, das Versionsverwaltungs-Tool erfasst alle Änderungen an den Dateien mit Zeitstempel und Benutzerkennung und verwaltet sie im sogenannten **Repository**. So wird gewährleistet, dass jeder mit dem aktuellen Repository

Stand arbeitet, bei Bedarf aber jederzeit auch auf einen älteren Stand zurückgreifen kann.

Die Versionsverwaltung bietet für die Softwareentwicklung folgende Vorteile:

- ▶ **Protokollierung der Änderungen:**
Jeder kann jederzeit nachvollziehen, wer für welche Änderung verantwortlich ist und wann die Änderung vorgenommen wurde.
- ▶ **Wiederherstellung älterer Versionen:**
Versehentliche Änderungen können so wieder rückgängig gemacht werden.
- ▶ **Koordinierung des gemeinsamen Zugriffs:**
Beim Update wird kontrolliert, ob es in der lokalen Arbeitskopie eine Version dieser Datei gibt, die noch nicht in das System übertragen und im gleichen Bereich geändert wurde. Wenn ja, macht das System darauf aufmerksam. Diese sogenannten Konflikte müssen manuell gelöst werden.
- ▶ **Archivierung von Releaseständen (Tags):**
Fertiggestellte Releases können als Gesamtheit archiviert und somit immer wieder geladen werden.
- ▶ **Entwicklung mehrerer Entwicklungszweige (Branches):**
Durch das Auslagern in einen Entwicklungsweig kann gleichzeitig an mehreren Releaseversionen gearbeitet werden. Beim Zusammenführen mit der Hauptversion unterstützt das System die Entwickler.

SVN Wir setzen für unsere Projekte die Versionsverwaltung Subversion (SVN) ein. Voraussetzung, um SVN nutzen zu können, ist ein SVN-Repository auf Ihrem Server. Im Abschnitt 18.2.4 auf Seite 663 wird gezeigt, wie ein Subversion-Repository auf einem Server installiert wird.

7.13.1 Ein Rails-Projekt in Subversion importieren

Nehmen wir an, Sie möchten ein Rails-Projekt mit dem Namen `railsair` erstellen und in Subversion importieren:

Wir erstellen das Projekt in einem temporären Verzeichnis (`tmp_svn`).

```
mkdir tmp_svn
cd tmp_svn
# Nur wenn Projekt noch nicht erstellt worden ist.
rails railsair
```

Falls Sie bereits vorher das Rails-Projekt generiert haben (siehe Kapitel »Test-Driven Development«), so verschieben Sie einfach Ihr Projekt in das temporäre Verzeichnis `tmp_svn`.

In Subversion ist es üblich, dass die Projekte folgende Verzeichnisstruktur besitzen:

Verzeichnis-
struktur

1. Trunk (dt. Stamm):

Hier befindet sich Ihr Rails-Projekt in der aktuellen Version.

2. Tags (dt. Markierung):

Hier befinden sich ältere, in sich abgeschlossene Versionen Ihres Projekts, die mit einem Namen gekennzeichnet sind (Releases, zum Beispiel: `version_1.0.4`)

3. Branches (dt. Verzweigungen):

Hier werden alternative Entwicklungspfade Ihres Projektes verwaltet.

Wir werden die Verzeichnisse `branches` und `tags` anlegen und das Verzeichnis des Rails-Projektes in `trunk` umbenennen:

```
mkdir branches
mkdir tags
mv railsair trunk
```

Bevor wir das Projekt in das Repository einchecken, löschen bzw. leeren wir überflüssige Log- und tmp-Dateien, da diese nicht im Repository benötigt werden. Dazu können wir die Rake-Tasks `rake tmp:clear` und `rake log:clear` verwenden:

```
cd trunk
rake tmp:clear
rake log:clear
```

Wir können nun mit dem Befehl `svn import` unser Projekt in das Repository überführen. Wechseln Sie dazu in das Verzeichnis `tmp_svn` (überprüfen Sie, ob Sie beim Ausführen des Befehls `ls` keine Fehlermeldung erhalten), und verwenden Sie den Befehl `svn import`, um das Projekt in das Subversion-Repository zu überführen.

svn import

```
cd ..
ls
  branches tags trunk
svn import url_des_repository -m 'Import des Railsprojekts'
```

Die URL des Repository kann z. B. wie folgt aussehen:

- ▶ **http://domain/railsair**
- ▶ **https://domain/railsair**
- ▶ **svn+ssh://deploy@domain/srv/svn/railsair**

Wenn Sie keine Fehlermeldung erhalten, können Sie das Verzeichnis `tmp_svn` löschen. Wir empfehlen jedoch, damit zu warten, bis der erste Check-out des Projektes aus dem Repository erfolgreich war.

7.13.2 Ein Rails-Projekt aus Subversion auschecken

`svn checkout` Wechseln Sie in das Verzeichnis, in dem Ihr Rails-Projekt liegen soll, und verwenden Sie den Subversion-Befehl `svn checkout`, um das Projekt auszuchecken. Achten Sie darauf, dass Sie das `trunk`-Verzeichnis angeben, da sich dort unsere Projektdateien befinden. Anschließend wechseln Sie in das Projektverzeichnis.

```
cd ~/railsprojekte
svn checkout url_des_repository/trunk railsair
cd railsair
```

7.13.3 Ignorieren von Dateien

`svn propset svn:ignore` Damit geänderte oder neue Dateien aus dem `tmp`- und `log`-Verzeichnis nicht in das Repository übertragen werden, müssen wir Subversion mitteilen, dass diese ignoriert werden sollen:

```
svn propset svn:ignore '*.log' log
svn propset svn:ignore '**' tmp/cache
svn propset svn:ignore '**' tmp/pids
svn propset svn:ignore '**' tmp/sessions
svn propset svn:ignore '**' tmp/sockets
```

Wenn Sie SQLite3-Datenbanken verwenden, können Sie die Dateien `db/*.sqlite3` ignorieren, da diese in der Regel nur temporär zur Entwicklung benötigt werden. Auf dem Server wird meistens MySQL, PostgreSQL oder ein anderes Datenbanksystem verwendet.

```
svn propset svn:ignore '*.sqlite3' db
```

Zuletzt können Sie noch einige Dateien bzw. Verzeichnisse im Verzeichnis `doc` ignorieren. Da diese sich gemeinsam mit nicht zu ignorierenden Dateien im selben Verzeichnis befinden, müssen Sie hier etwas anders vorgehen. Wir verwenden den Befehl `svn probedit`, der ein Editor-Fenster

(in der Regel der vi-Editor) öffnet, in dem Sie pro Zeile eine zu ignorierende Datei angeben können.

```
svn propedit svn:ignore doc
# folgende Zeilen im Editor eingeben
# (im VI-Editor i eingeben zum Starten der Eingabe)
app
api
# anschließend speichern und Editor verlassen
# (ESC :wq beim VI-Editor)
```

Mit dem Befehl `svn commit` übertragen wir diese Änderungen in das Subversion-Repository: `svn commit`

```
svn commit -m "ignoriere Dateien"
```

Über den Parameter `-m` geben wir an, welche Änderungen mit einem `commit` übertragen wurden. Dadurch ist es uns später immer wieder möglich nachzuvollziehen, was bei dieser Änderung geändert wurde.

Ignorieren der `database.yml`

[+]

Aus Sicherheitsgründen kann es sinnvoll sein, die Datenbankkonfigurationsdatei **database.yml** nicht mit in das Repository zu übertragen. Stattdessen können Sie eine Beispieldatei mit dem Namen **database.yml.example** erstellen und diese übertragen. Da am Anfang die **database.yml** noch keine geheimen Daten enthält, können Sie diese dafür umbenennen. Außerdem müssen Sie noch festlegen, dass die Datenbankkonfigurationsdatei ignoriert werden soll:

```
svn mv config/database.yml config/database.yml.example
svn propset svn:ignore "database.yml" config/
svn commit config -m "database.yml wird ignoriert."
```

Denken Sie daran, beim Auschecken des Projektes aus dem Subversion-Repository die Datei `database.yml` wiederherzustellen. Wenn Sie das Projekt mit dem Deployment-Tool Capistrano auf Ihrem Webserver installieren, ist es notwendig, dass ein entsprechender Deploy-Task diesen Vorgang automatisiert durchführt. Siehe Kapitel 18.

Index

A

- Absendebutton 361
- Acceptance-Test 279
- Action-Caching 631
- ActionController
 - after_filter* 415
 - around_filter* 416
 - Aufgaben des Controllers 401
 - Authentifizierung 413
 - before_filter* 414
 - Benannte Routen 420
 - Cookies 404
 - Filter 414
 - Flash-Nachrichten 408
 - Grundlagen 399
 - params[]* 401
 - render* 406
 - request* 402
 - respond_to* 407
 - Ressourcen 424
 - Routen mit Regulären Ausdrücken 422
 - Routing 417
 - Routing-Regeln 419
 - send_data* 412
 - send_file* 411
 - Sessions 405
 - Weiterleitungen 409
- ActionMailer 523
- ActionView
 - Alternative Template-Systeme 396
 - Formulare mit Bezug zu einem Model 353
 - Formulare mit Bezug zu mehr als einem Model 385
 - Formulare ohne Bezug zu einem Model 387
 - Helper 329
 - Layouts 351
 - Partials 389
 - Templates 325
- ActiveRecord 175
 - :conditions* 453
 - :limit* 456
 - :offset* 456
 - :order* 455
- Assoziationen 475
- Assoziationen mit Bedingungen 499
- Assoziationen um eigene Methoden erweitern 502
- Callbacks 516
 - create* 447
 - delete* 449
 - destroy* 448
 - Einführung 425
 - Eins-zu-eins-Assoziation 484
 - Eins-zu-viele-Assoziation 475
 - find* 449
 - find_by_feldname* 457
 - find_by_sql* 458
 - freeze* 448
 - Getter- und Setter-Methoden 444
 - Join-Tabelle 488
 - Mehrere Assoziationen zur selben Tabelle 498
 - Migration-Generator 461
 - Migrations 459
 - Model generieren 435
 - new* 446
 - Polymorphe Assoziationen 494
 - rake db:migrate* 466
 - Rake-Tasks 440
 - self.down* 465
 - self.up* 465
 - Single Table Inheritance 519
 - Sortierreihenfolge 455
 - Statistische Berechnungen 515
 - Suchen 449
 - Suchoptionen 452
 - update* 448
 - update_attribute* 447
 - validates_acceptance_of* 503
 - validates_confirmation_of* 505
 - validates_each* 514
 - validates_exclusion_of* 506
 - validates_format_of* 508
 - validates_inclusion_of* 507
 - validates_length_of* 508
 - validates_numericality_of* 511
 - validates_presence_of* 512
 - validates_size_of* 508

- validates_uniqueness_of* 513
- validation_associated* 504
- Validierung* 503
- Vererbung* 518
- Viele-zu-viele-Assoziation* 487
- ActiveResource 587
- ActiveSupport 539
- ActiveSupport-Methoden
 - ago* 544
 - assert_valid_keys* 548
 - at* 545
 - beginning_of_month* 541
 - beginning_of_quarter* 541
 - beginning_of_week* 541
 - blank?* 548
 - byte* 541
 - day* 543
 - diff* 547
 - end_of_month* 542
 - even?* 540
 - exabyte* 541
 - except* 548
 - first* 545
 - fortnight* 543
 - from* 546
 - from_now* 543
 - gigabyte* 541
 - hour* 543
 - in_groups_of* 546
 - kilobyte* 541
 - last* 545
 - last_month* 542
 - last_year* 542
 - megabyte* 541
 - minute* 543
 - months_ago* 542
 - months_since* 542
 - multiple_of?* 540
 - next_month* 542
 - odd?* 540
 - ordinalize* 540
 - petabyte* 541
 - rand* 547
 - reverse_merge* 548
 - second* 543
 - since* 543
 - split* 547
 - stringify_keys* 547
 - symbolize_keys* 547
 - terabyte* 541
 - to* 546
 - to_date* 544
 - to_json* 549
 - to_options* 547
 - to_param* 546
 - to_sentence* 546
 - to_time* 544
 - to_yaml* 549
 - today* 544
 - tomorrow* 544
 - until* 544
 - week* 543
 - year* 543
 - years_ago* 542
 - years_since* 542
 - yesterday* 544
- Acts as List 616
- Acts as Paranoid 616
- Acts as State Machine 616
- Acts as Taggable on Steroids 616
- Acts as Versioned 616
- after_filter* 415
- ago* 544
- Ajax 232
 - assign* 567
 - Beispiel* 559
 - call* 567
 - delay* 568
 - Einbinden der JavaScript-Dateien* 554
 - Firebug* 561
 - form_remote_tag* 557
 - Grundlagen* 551
 - hide* 564
 - insert_html* 566
 - link_to_function* 558
 - link_to_remote* 557
 - observe_field* 558
 - page.alert* 567
 - periodically_call_remote* 558
 - Prototype* 552
 - redirect_to* 567
 - remote_form_for* 557
 - remove* 567
 - replace* 565
 - replace_html* 565
 - RJS – Ruby-JavaScript* 555
 - RJS-Referenz* 564
 - script.aculo.us* 552
 - show* 564

- toggle* 564
- visual_effect* 564
- Alternative Template-Systeme 396
- Apache-Webserver 667
- around_filter* 416
- Array-Methoden
 - in_groups_of* 546
 - rand* 547
 - split* 547
 - to_param* 546
 - to_sentence* 546
- Arrays 546
 - auf Arrayelemente zugreifen* 117
 - clear* 127
 - compact* 126
 - delete* 126
 - delete_at* 126
 - delete_if* 127
 - detect* 124
 - find* 124
 - find_all* 124
 - first* 119
 - grep* 124
 - last* 119
 - length* 120
 - max* 124
 - min* 124
 - new* 117
 - nitems* 120
 - pop* 127
 - rand* 123
 - reject* 124
 - reverse* 127
 - select* 124
 - shift* 127
 - size* 120
 - sort* 121
 - sort_by* 122
 - sortieren* 121
 - uniq* 128
 - values_at* 119
 - vergleichen* 120
 - zufällig sortieren* 123
- assert* 248, 284
- assert_difference* 271, 284
- assert_equal* 251, 284
- assert_instance_of* 284
- assert_not_nil* 270, 284
- assert_redirected_to* 271, 285
- assert_respond_to* 260, 284

- assert_response* 270, 285
- assert_select* 275, 284
- assert_valid_keys* 547, 548
- Assoziationen 475
- Assoziationen mit Bedingungen 499
- Assoziationen um eigene Methoden
 - erweitern 502
- at* 545
- Attachment Fu 617
- Authentifizierung 413
 - HTTP-Authentifizierung* 64
 - Plug-in* 597
- Authentifizierungssystem 206
- Autotest 282

B

- before_filter* 414
- beginning_of_month* 541
- beginning_of_quarter* 541
- beginning_of_week* 541
- Benannte Routen 420
- blank?* 548
- button_to* 332
- byte* 541

C

- Cachen der Root-Page 629
- Caching 619
- Caching mit memcached 642
- Caching von CSS- und JavaScript-Dateien 640
- Caching-Einstellungen 622
- Callbacks 516
- Capistrano 673
- capitalize* 98
- center* 97
- Checkboxen 364
- chomp* 103
- chop* 103
- concat* 102
- conditions* 453
- content_tag* 348
- Controller
 - respond_to* 556
- Controller-Generator 156
- Cookies 404
- count* 108
- create* 447

Cross-Site Request Forgery 650
 Cross-Site-Scripting 648
 crypt 107
 CSRF → Cross-Site Request Forgery
 cycle 347

D

Date 541
 Date-Methoden
 beginning_of_month 541
 beginning_of_quarter 541
 beginning_of_week 541
 end_of_month 542
 last_month 542
 last_year 542
 months_ago 542
 months_since 542
 next_month 542
 today 544
 tomorrow 544
 years_ago 542
 years_since 542
 yesterday 544
 Datei-Upload 363
 Datenbank-Konfiguration 296
 Datenbankpersistenz 31
 Datentyp-Klassen
 Arrays 546
 Date 541
 DateTime 541
 Hashes 547
 Integer 540
 Numeric 540
 String 545
 Time 541
 Zahlen 540
 Zeichenketten 545
 DateTime 541
 Datum
 formatieren 276
 Datum und Zeit 541
 Datumsauswahllisten 367
 day 543
 debug 349
 Debugging 311
 delete 449
 Deployment
 Capistrano 673
 Einrichten des Servers 659

*Konfigurieren der Rails-
 Applikation* 669
Subversion 663
Wahl des Providers 658
 destroy 448
 diff 547
 downcase 98
 DRY-Prinzip 32

E

E-Mail
 ActionMailer 523
 Anhänge 536
 Beispielprojekt Kontaktformular 523
 HTML-E-Mails 535
 Konfiguration 536
 sendmail 536
 SMTP 536
 EdgeRails 319
 Editoren 55
 Eigene Helper entwickeln 350
 Eine einfache Bookmark-
 verwaltung 153
 Eins-zu-eins-Assoziation 484
 Eins-zu-viele-Assoziation 475
 Einzeilige Textfelder 357
 end_of_month 542
 Entwicklungsumgebungen → Editoren
 eql? 82
 equal? 81
 Error 273
 Erste Schritte 59
 even? 540
 exabyte 541
 except 548
 Exception Notifier 615
 excerpt 342

F

Failure 273
 Fehlerbenachrichtigung
 Plug-in 615
 Felder 546
 Filter 414
 find 449
 find_by_feldname 457
 find_by_sql 458
 Firebug 616

first 545
 Fixtures 250, 274
 Flash-Messages 197
 Flash-Nachrichten 408
 Formulare 185

- Absendebutton* 361
- Checkboxes* 364
- Datei-Upload* 363
- Datumsauswahllisten* 367
- Einzeilige Textfelder* 357
- Länderauswahllisten* 370
- Label* 362
- Listfelder* 372, 375
- Mehrzeilige Textfelder* 358
- Mit Bezug zu einem Model* 353
- Mit Bezug zu mehr als einem Model* 385
- Ohne Bezug zu einem Model* 387
- Passworteingabefelder* 359
- Pflichtfelder* 252
- Radiobuttons* 366
- Select-Felder* 277
- Validierung* 379
- Versteckte Felder* 360
- Zeit- und Datumsauswahllisten* 370
- Zeitauswahllisten* 369

 fortnight 543
 Fragment-Caching 635
 freeze 448
 Fremdschlüssel 244
 from 546
 from_now 543
 Functional-Test 242, 269

G

Generatoren 304

- Controller-Generator* 156
- Mailer* 526
- Model-Generator* 172
- Scaffold* 61, 230, 264

 Generieren eines Rails-Projektes 289
 Getter- und Setter-Methoden 444
 gigabyte 541
 Globalite 604
 Gravatar 614

H

Hash-Methoden

- assert_valid_keys* 548
- diff* 547
- except* 548
- reverse_merge* 548
- stringify_keys* 547
- symbolize_keys* 547
- to_options* 547

 Hashes 547

- clear* 134
- default* 131
- delete* 134
- delete_if* 135
- detect* 138
- each* 135
- each_key* 135
- each_pair* 135
- each_value* 135
- empty?* 136
- erzeugen* 130
- fetch* 132
- find* 138
- find_all* 138
- has_key?* 136
- has_value?* 137
- include?* 136
- invert* 136
- key?* 136
- keys* 137
- length* 137
- member?* 136
- merge* 139
- reject* 135
- select* 138
- shift* 134
- size* 137
- sort* 138
- sortieren* 138
- to_a* 137
- update* 139
- value?* 137
- values* 137
- values_at* 137

 Helper 200, 255

- button_to* 332
- content_tag* 348
- cycle* 347
- debug* 349

Eigene Helper entwickeln 350
excerpt 342
highlight 342
image_tag 335
javascript_include_tag 338
link_to 330
mail_to 333
markdown 345
number_to_currency 340
number_to_human_size 341
number_to_percentage 341
number_to_phone 342
number_with_delimiter 339
number_with_precision 340
pluralize 344
reset_cycle 348
sanitize 346
strip_links 347
strip_tags 347
stylesheet_link_tag 167, 337
textilize 346
textilize_without_paragraph 346
truncate 343
word_wrap 344
 here-document 93
 highlight 342
 hour 543
 HTTP-Authentifizierung 64
 HTTP-Methoden 572

I

i18n → Internationalisierung
image_tag 335
in_groups_of 546
 Installation
 Unter Linux 53
 Unter Mac OS X 42
 Unter Windows 48
 Integer 540
 Integer-Methoden 540
 even? 540
 multiple_of? 540
 odd? 540
 ordinalize 540
 Integration-Test 242, 279
 Interaktive Ruby Shell 69
 Internationalisierung
 Grundlagen 604
 Plug-in 604

irb → Interaktive Ruby Shell
 Iteratoren
 all? 130
 any? 130
 each 128
 each_index 129
 map 129
 reverse_each 128

J

JavaScript 551
javascript_include_tag 338, 554
 Join-Tabelle 488

K

Kapazitätseinheiten 541
 kilobyte 541
 Konfigurieren der Rails-Applikation
 auf dem Server 669
 Konsole 176, 308
 Konvention statt Konfiguration 31

L

Länderauswahllisten 370
 Label 362
 last 545
 last_month 542
 last_year 542
 Layout 165
 Layout-Partials 396
 Layouts 351
 Leerräume 104
link_to 330
 Listfelder mit dynamischen
 Werten 375
 Listfelder mit statischen Werten 372
 ljust 97
 Load Balancing 659
 Logging 310
 Lokaler Rails-Server 154
 Lokaler Server 309
 lstrip 104

M

mail_to 333
 Mailer 526
 markdown 345
 megabyte 541
 Mehrere Assoziationen zur selben
 Tabelle 498
 Mehrsprachigkeit → Internationalisierung
 Mehrzeilige Textfelder 358
 memcached 642
 Migration-Generator 461
 Migrations 173, 459
 minute 543
 Mixins 148
 Model
 Relationen 260
 Model generieren 435
 Model View Controller 30
 Model-Generator 172
 Module 147
 modulo 540
 Mongrel 666
 Mongrel-Cluster 666
 months_ago 542
 months_since 542
 multiple_of? 540
 MySQL
 Datenbank anlegen 664
 Installation 664

N

Namenskonventionen 294
 Namensräume 147
 Namespaces 577
 Neues in Rails 2.0.x 33
 new 446
 next_month 542
 Nginx 660
 Nomen 571
 number_to_currency 340
 number_to_human_size 341
 number_to_percentage 341
 number_to_phone 342
 number_with_delimiter 339
 number_with_precision 340
 Numeric 540
 Numeric-Methoden
 byte 541

day 543
exabyte 541
fortnight 543
gigabyte 541
hour 543
kilobyte 541
megabyte 541
minute 543
petabyte 541
second 543
terabyte 541
week 543
year 543

O

odd? 540
 ordinalize 540
 Ordinalzahlen 540

P

Page-Caching 620
 params[] 401
 Partial 203
 Layout-Partials 396
 Parameter :locals 391
 Partials mit Ressourcen 392
 Shared Partials 394
 Passworteingabefelder 359
 Performance
 Action-Caching 631
 Cachen der Root-Page 629
 Caching 619
 Caching mit memcached 642
 Caching von CSS- und JavaScript-
 Dateien 640
 Caching-Einstellungen 622
 Fragment-Caching 635
 Page-Caching 620
 petabyte 541
 Pflichtfelder definieren 178
 Pflichtfelder 252
 PHP 545
 Plug-ins
 Acts as List 616
 Acts as Paranoid 616
 Acts as State Machine 616
 Acts as Taggable on Steroids 616
 Acts as Versioned 616

- Attachment Fu* 617
 - Exception Notifier* 615
 - Globalite* 604
 - Gravatar* 614
 - Grundlagen* 593
 - Quellen* 596
 - Restful Authentication* 597
 - Will Paginate* 617
 - pluralize 344
 - Polymorphe Assoziationen 494
 - Prototype 552
- ## R
-
- Radiobuttons 366
 - Rails
 - Datenbank-Konfiguration* 296
 - Datenbankpersistenz* 31
 - Debugging* 311
 - DRY* 32
 - EdgeRails* 319
 - Editoren* 55
 - Generatoren* 304
 - Generieren eines Rails-Projektes* 289
 - Installation* 41, 665
 - Konvention statt Konfiguration* 31
 - Lokaler Server* 309
 - Namenskonventionen* 294
 - Neues in Rails 2.0.x* 33
 - Rails 2.0.2* 36
 - Rails-Konsole* 308
 - Rake* 314
 - Umgebungseinstellungen* 299
 - Update auf Rails 2.0.x* 36
 - Verzeichnisstruktur einer Rails-Applikation* 292
 - Warum Ruby?* 28
 - Wie entstand Rails?* 27
 - Rails 2.0.2 36
 - Rails-Console 539
 - Rails-Konsole 176, 308
 - Rake 314
 - rake db:migrate 466
 - Rake-Tasks 283
 - Datenbanken* 440
 - Fixtures* 442
 - fixtures:load* 283
 - Migrations* 441
 - Schema-Dateien* 443
 - Sessions* 443
 - test* 283
 - test:functionals* 283
 - test:integration* 283
 - test:plugins* 283
 - test:recent* 283
 - test:uncommitted* 283
 - test:units* 283
 - Testdatenbank* 442
 - rand 547
 - redirect_to 409
 - Refaktorisierung 200
 - Reguläre Ausdrücke 111
 - Relationen 244, 260, → Assoziationen
 - render 406
 - request 402
 - reset_cycle 348
 - respond_to 407, 556
 - Ressourcen
 - erweitern* 582
 - Singleton* 579
 - verschachtelt* 576
 - REST-Standard 570
 - Restful Authentication 597
 - RESTful Rails 223, 573
 - reverse 108
 - reverse_merge 548
 - RJS – Ruby-JavaScript 555
 - RJS-Referenz 564
 - rjust 97
 - root-Route 420
 - Routen mit Regulären Ausdrücken 422
 - Routing 216, 417
 - Routing-Regeln 419
 - RSpec 243
 - rstrip 104
 - Ruby
 - Arrays* 116
 - capitalize* 98
 - case* 80
 - center* 97
 - chomp* 103
 - chop* 103
 - concat* 102
 - count* 108
 - crypt* 107
 - Datum und Zeit* 139
 - delete* 109
 - downcase* 98
 - dump* 110

- eql?* 82
- equal?* 81
- Hashes* 130
- here-document* 93
- if* 78
- include* 101
- index* 101
- Installation* 663
- Interaktive Ruby Shell* 69
- irb* 69
- Iteratoren* 128
- Klassen* 84
- ljust* 97
- lstrip* 104
- Mehrfachverzweigungen* 80
- Mixins* 148
- Module* 147
- Namensräume* 147
- Objekte und Datentypen* 73
- Reguläre Ausdrücke* 111
- reverse* 108
- rindex* 101
- rjust* 97
- rstrip* 104
- scan* 101
- Schleifen* 82
- sprintf* 97
- squeeze* 109
- strip* 104
- swapcase* 98
- Symbole* 110
- Time* 139
- to_f* 105
- to_i* 105
- Try Ruby* 70
- until-Schleife* 82
- upcase* 98
- Variablen* 72
- Verzweigungen* 78
- Was ist Ruby?* 67
- while-Schleife* 82
- Zahlen* 88
- Zeichenketten* 91
- RubyGems
 - Installation* 663
- Scaffold-Generator 230, 264
- script.aculo.us 552
- script/console 539
- second 543
- Select-Felder 277
- Selenium 279
- self.down 465
- self.up 465
- send_data 412
- send_file 411
- sendmail 536
- Server
 - Webserver Nginx* 660
 - Apache-Webserver* 667
 - Installation* 659
 - Load Balancing* 659
 - Mongrel* 666
 - MySQL* 664
 - Root-User* 660
 - Subversion* 663
 - Ubuntu* 660
 - Virtualisierung* 660
- Session Fixation 652
- Session Hijacking 652
- Sessions 405
- Shared Partials 394
- Sicherheit
 - Cross-Site-Request-Forgery* 650
 - Cross-Site-Scripting* 648
 - Reguläre Ausdrücke* 655
 - Session Fixation* 652
 - Session Hijacking* 652
 - SQL Injection* 647
 - Validierung* 654
 - Warum Sicherheit wichtig ist* 647
- since 543
- Single Table Inheritance 519
- Singleton-Ressourcen 579
- SMTP 536
- Sortierreihenfolge 455
- sources.list 661
- split 547
- sprintf 97
- SQL Injection 647
- SQLite
 - Installation* 665
- squeeze 109
- Statistische Berechnungen 515
- Story 280
- String-Methoden

S

sanitize 346
Scaffold 61

- at* 545
- fiest* 545
- from* 546
- last* 545
- to* 546
- stringify_keys 547
- strip 104
- strip_links 347
- strip_tags 347
- stylesheet_link_tag 167, 337
- Subversion 663
- Suchoptionen 452
- SVN-Repository 663
- swapcase 98
- Sweeper 627
- Symbole 110
- symbolize_keys 547

T

- TDD → Testen
- Teilstrings 99
- Templates 325
- terabyte 541
- Test-Driven Development → Testen
- Testbefehle
 - assert* 248, 284
 - assert_difference* 271, 284
 - assert_equal* 251, 284
 - assert_instance_of* 284
 - assert_not_nil* 270, 284
 - assert_redirected_to* 271, 285
 - assert_respond_to* 260, 284
 - assert_response* 270, 285
 - assert_select* 275, 284
- Testen
 - Acceptance-Test* 279
 - Autotest* 282
 - Fixtures* 250
 - Functional-Test* 242, 269
 - Grundlagen* 241
 - Helper* 255
 - Integration-Test* 242, 279
 - mit Selenium* 279
 - mit TextMate* 249
 - RSpec* 243
 - Story* 280
 - Unit-Test* 242
 - ZenTest* 282
- Testgetriebene Progr. → Testen

- Textfelder
 - Einzeilige* 357
 - Mehrzeilige* 358
- textilize 346
- textilize_without_paragraph 346
- Time 541
 - at* 142
 - gregorian_leap?* 144
 - julian_leap?* 144
 - local* 140
 - mktime* 140
 - parsedate* 146
 - Schaltjahre bestimmen* 144
 - to_f* 142
 - to_i* 142
 - utc* 140
 - vergleichen* 145
 - wday* 141
 - Wochentag bestimmen* 141
 - yday* 142
- to 546
- to_date 544
- to_json 549
- to_options 547
- to_param 546
- to_sentence 546
- to_time 544
- to_yaml 549
- today 544
- tomorrow 544
- truncate 343
- Try Ruby 70

U

- Ubuntu 660
- Umgebungseinstellungen 299
- Unit-Test 242, 246
- until 544
- until-Schleife 82
- upcase 98
- update 448
- Update auf Rails 2.0.x 36
- update_attribute 447

V

- validates_acceptance_of 503
- validates_confirmation_of 505
- validates_each 514

validates_exclusion_of 506
 validates_format_of 508
 validates_inclusion_of 507
 validates_length_of 254, 508
 validates_numericality_of 511
 validates_presence_of 178, 253, 512
 validates_size_of 508
 validates_uniqueness_of 513
 validation_associated 504
 Validierung 379, 503
 Verben 572
 Vererbung 518
 Verschachtelte Ressourcen 576
 Versteckte Felder 360
 Verzeichnisstruktur einer Rails-Applikation 292
 Viele-zu-viele-Assoziation 487
 View erstellen 159
 Virtualisierung 660

W

Wahl des Providers 658
 Warum Ruby? 28
 Was ist Ruby? 67
 Webservice → Server
 Webservice anbieten 584
 Webservices 569
 week 543
 Weiterleitungen 409
 while-Schleife 82
 Whitespaces 104
 Wie entstand Rails? 27
 Will Paginate 617

word_wrap 344

X

XML 551
 XSRF → Cross-Site Request Forgery
 XSS → Cross-Site-Scripting

Y

YAML 251
 year 543
 years_ago 542
 years_since 542
 yesterday 544

Z

Zahlen 88, 540
 Zeichenketten 91

- formatieren* 97
- Groß- und Kleinschrift* 98
- in Zahlen konvertieren* 105
- length* 95
- split* 96
- suchen* 101
- Teilstrings* 99
- umkehren* 108
- verschlüsseln* 107
- Zeichen zählen* 108

 Zeit- und Datumsauswahllisten 370
 Zeitauswahllisten 369
 ZenTest 282