entwickler.press

Ruby on Rails 3

Data Mapper • HAML und SASS • Release- und Sourcecode-Management mit Capistrano und Git • Test Driven Development (TDD) • Volltextsuche mit Sphinx

Michael Voigt, Patrick Hesse

Michael Voigt, Stefan Tennigkeit

Ruby on Rails 3

Michael Voigt, Stefan Tennigkeit

Ruby on Rails 3

DataMapper | Haml und Sass | RubyGems & Bundler | Capistrano | Test Driven Development (TDD) | Volltextsuche mit Sphinx | I18N & L10N

entwickler.press

Michael Voigt, Stefan Tennigkeit
Ruby on Rails 3
ISBN: 978-3-86802-229-2

Ein Imprint der Software & Support Media GmbH

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Ihr Kontakt zum Verlag und Lektorat:
Software & Support Media GmbH
entwickler.press
Geleitsstr. 14
60599 Frankfurt am Main
Tel.: +49 (0)69 630089 0
Fax: +49 (0)69 930089 89
lektorat@entwickler-press.de
http://www.entwickler-press.de

Lektorat: Sebastian Burkart
Korrektorat: Frauke Pesch
Satz: Pobporn Fischer
Belichtung, Druck & Bindung: M.P. Media-Print Informationstechnologie GmbH, Paderborn

Inhaltsverzeichnis

Vorwort

Ruby on Rails (kurz Rails) ist ein sehr bekanntes MVC-Framework im Bereich der Webentwicklung. Einer der Hauptgründe hierfür ist sicher das bequeme und schnelle Entwickeln von Prototypen. Aber auch über den Prototypen hinaus ist Rails für die Erstellung großer und komplexer Applikationen geeignet. Viele Nutzer sprechen nach erstmaligem Kontakt mit Rails von einem gewissen Spaßfaktor mit Hinblick auf das schnelle Feedback durch funktionierenden Code bei der Entwicklung einer Applikation.

Somit ist es auch nicht verwunderlich, dass fast jeder Verlag von IT-Büchern ein oder mehrere Rails-Bücher in seinem Sortiment vorweisen kann. Nun stellen sich berechtigterweise folgende Fragen: *Welche Motivation haben die Autoren, ein weiteres Rails-Buch zu schreiben?* oder *Was unterscheidet dieses Buch von anderen Rails-Büchern?* Zu Beginn war uns nicht bewusst, dass wir ein Rails-Buch schreiben würden, da wir zunächst die Absicht hatten, ein Buch über das Webframework Merb zu schreiben, welches ebenfalls in der Programmiersprache Ruby geschrieben ist und von Rails stark beeinflusst wurde. Am 23. Dezember 2008 gab David Heinemeier Hansson bekannt, dass die besten Implementierungen von Merb und Rails zu Rails 3 zusammengefügt werden sollen. So entstand in Absprache mit unserem Verlag die Idee, ein Rails-3-Buch zu schreiben, da bis zu diesem Zeitpunkt die Weiterentwicklung von Merb ungewiss war. Nun stellten auch wir uns die Frage, wie wir den Lesern die verschiedenen Themengebiete von Rails und der Webentwicklung am besten vermitteln könnten. Bei unseren Überlegungen fiel uns auf, dass sich die bestehenden Rails-Bücher in zwei Fraktionen aufspalteten. Zum einen gab es Bücher über Rails für Anfänger, in denen nur die Grundlagen vermittelt wurden. Zum anderen gab es den Bereich der so genannten Rezeptbücher, in denen konkrete Probleme mithilfe von Rails gelöst wurden. Wir hingegen wollen diese zwei Teilgebiete nicht wild mischen, sondern beide Aspekte innerhalb unseres Buches in zwei getrennten Abschnitten berücksichtigen. In den ersten Kapiteln geben wir einen Überblick über Rails und seine verschiedenen Komponenten. Im zweiten Teil des Buches, dem „Werkzeugkasten", wollen wir auf konkrete Herausforderungen der alltäglichen Softwareentwicklung eingehen. Die Beispiele und der jeweilige Quelltext wurden aus einer bestehenden Applikation (*http://www.kraeftemessen.com*) extrahiert und der entsprechende Quelltext ist einsehbar (*http://github.com/spider-network/kraeftemessen*).

Die Idee und somit auch die Motivation hinter dem Buch bestehen darin, die Grundlagen von Rails zu vermitteln und bestehende Lösungsansätze für konkrete Herausforderungen anzubieten. Wir hoffen, dass wir dieses ehrgeizige Ziel bereits in der ersten Auflage dieses Buches erreichen können. Da uns dies sicher nicht an jeder Stelle zu 100 % gelingen wird, bitten wir Sie um Ihr Feedback. Sie erreichen uns über die E-Mail-Adresse railsbook@spider-network.net. Nur mithilfe Ihrer Unterstützung werden wir etwaige Fehler verbessern und somit ein solides Rails-Buch auf die Beine stellen.

Wir Autoren und alle Mitwirkenden wünschen Ihnen viel Spaß beim Lesen und hoffen, dass Sie mithilfe des Buches einige Ihrer Programmierherausforderungen leichter bewältigen können.

Autoren und Danksagung

A.1 Michael Voigt (Autor)

Michael Voigt entwickelt seit mehr als fünf Jahren professionell Software, primär im Webbereich und der Integration von externen Systemen. Begonnen hat er als Webentwickler in Salvador, Brasilien bei der Firma viventura.de. Anschließend kehrte er zurück in seine Heimatstadt Leipzig und arbeitete bei der Firma LSL Consulting GmbH, wo er für die bedarfsgerechte Anpassung der Kundenportale im B2B-Bereich sowie deren Modularisierung verantwortlich war. Derzeit ist Michael Voigt als Softwareentwickler bei der wunderloop media services GmbH in Hamburg angestellt. Hier ist er sowohl für das Release- und Sourcecode-Management als auch für die vollständige Implementierung des Abrechnungssystems und dessen SAP-Integration zuständig. Neben der Arbeit als Softwareentwickler ist er ein begeisterter Sportler, der seine Herausforderung im Langstreckenschwimmen und im Triathlon sucht. Weitere Informationen zu Michael Voigt können Sie auf seiner Webseite spider-network.net finden.

Ich möchte mich an dieser Stelle bei den Menschen bedanken, die mich während des Verfassens des Buches und darüber hinaus unterstützt haben. Als Erstes bedanke ich mich bei meiner Freundin Sandra G., die stets Verständnis gezeigt hat, auch wenn ich nicht so viel Zeit hatte. Danke nochmals, dass du dir die einzelnen Kapitel angehört hast. Ich kann verstehen, dass dies nicht immer einfach war, wenn man selbst nicht aus dem Bereich der Informatik kommt. Durch dein Feedback hat das Buch sehr viel an Verständlichkeit gewonnen. Weiterhin möchte ich meiner Familie danken, dass sie mir immer alle Wege offen gehalten und mich stets bei meinen Vorhaben unterstützt hat. Ebenso möchte ich mich speziell bei Stefan Tennigkeit (Co-Autor) bedanken, denn ohne sein Engagement und sein schnelles und konstruktives sowie kritisches Feedback wäre das Buch nicht das, was es jetzt ist. Weiterhin möchte ich mich aber bereits jetzt bei den Menschen bedanken, die mir in Zukunft neue Herausforderungen stellen werden, denn nur über diese bleibt das Leben und die Arbeit spannend. Scheuen Sie sich nicht, mich zu kontaktieren (challenges@spider-network.net). Ich warte auf Ihre Herausforderungen.

A.2 Stefan Tennigkeit (Koautor)

Stefan Tennigkeit absolvierte sein Informatikstudium an der FH Merseburg. Während des Studiums arbeitete er am Max-Planck-Institut für Kognitions- und Neurowissenschaften in Leipzig. Dort arbeitete er an der internen Informationsplattform, basierend auf TWiki. Seit Anfang 2008 ist er als Ruby-/Rails-Entwickler bei der wunderloop media services GmbH in Hamburg im Bereich Implementierung der Abrechnung und Integration mit SAP sowie der Weiterentwicklung der Behavioral-Targeting-Marktplatzplattform „connect“ tätig.

1 Einstieg

Herzlich willkommen liebe Leserinnen und Leser,

wir freuen uns, dass Sie sich für dieses Rails-Buch entschieden haben. Es beinhaltet eine vollständige Einführung in die Entwicklung von Webapplikationen mit Ruby on Rails (kurz Rails) und gibt Ihnen konkrete Hinweise, wie Sie Herausforderungen in der Webentwicklung lösen können. Rails hat mittlerweile schon einige Jahre auf dem Buckel, seit es am 13. Dezember 2005 in der Version 1.0 als Public-Release das Licht der Software-Welt erblickte. Innerhalb dieser Zeit hat sich das Framework enorm weiterentwickelt, wie auch die Webentwicklung an sich. Durch das geschickte Marketing von Rails ist das Framework den meisten Webentwicklern bereits ein Begriff, selbst wenn sie nicht direkt mit Rails arbeiten. Rails inspirierte ebenso Entwickler abseits der Programmiersprache Ruby. Hierbei entstanden Webframeworks, die Rails sehr ähnlich sind, beispielsweise CakePHP oder Grails aus der Java-Welt. Durch die große Community von Rails entwickelt sich das Framework stetig weiter. Rails ist bequem einsetzbar, um kleine Webseiten oder Portale zu entwickeln, aber auch für die Entwicklung geschäftskritischer Anwendungen verwendbar. Mit diesem Buch möchten wir Ihnen zeigen, wie Sie mit Rails Webanwendungen entwickeln können. Wenn Sie bereits Erfahrungen mit Rails sammeln konnten, zeigen wir Ihnen außerdem, was sich in der Version 3.0 von Rails verändert hat.

1.1 Zielgruppe des Buches

Dieses Buch ist für Einsteiger geeignet, die noch keine Erfahrungen mit Rails haben. Aber auch Entwickler, die bereits Rails-Applikationen entwickelt haben, werden auf ihre Kosten kommen. Wollen Sie gerade mit einem IT-Studium/-Ausbildung beginnen oder stecken gar mitten drin? Möchten Sie Rails 3 in einem Projekt einsetzen? Oder haben Sie bereits Applikationen mit Rails 2 entwickelt? Dann ist dieses Buch genau richtig, um Sie an das Thema „Rails“ heranzuführen oder bereits vorhandenes Wissen zu vertiefen. In diesem Buch setzen wir allerdings voraus, dass Sie ein Grundverständnis für die Technologie des Internets mitbringen und wissen, wie man unter Anleitung ein Programm installiert. Vorteilhaft wäre ebenfalls, wenn Sie bereits Erfahrungen mit einer Programmiersprache, speziell mit einer Skriptsprache wie zum Beispiel PHP, Perl oder Python haben. Grundlagenwissen in der Programmiersprache Ruby wäre optimal, wird für das Verständnis des Buches aber nicht zwingend vorausgesetzt, da alles Nötige Stück für Stück erklärt wird.

1.2 Informationen/Feedback

Die Webseite zu diesem Buch finden Sie unter *http://rails.spider-network.net*. Auf dieser Webseite finden Sie Korrekturen und weiterführende Informationen zu diesem Buch sowie zum Thema Rails selbst. Bitte senden Sie uns Ihre Anmerkungen, Hinweise und Korrekturvorschläge an rails-book@spider-network.net. Informationen zum Verlag und dessen Verlagsprogramm finden Sie unter *http://www.entwickler-press.de*.

1.3 Aufbau des Buches

Hier möchten wir Ihnen einen kurzen Ausblick über den Inhalt und den Aufbau des Buches geben. Die Besonderheit des Buches besteht darin, dass die meisten Quelltextbeispiele einer laufenden Applikation (*http://www.kraeftemessen.com*) entnommen wurden und ihr gesamter Quelltext auf Github (*http://github.com/spider-network/kraeftemessen*) einsehbar ist. Bei den jeweiligen Kapiteln wird diese Applikation als Leitfaden für die Erläuterung der verschiedenen Themengebiete verwendet. Bei der Applikation handelt es sich um eine Veranstaltungsdatenbank für sportliche Ausdauerwettkämpfe, mit der Besucher nach Veranstaltungen aus Ihrer Umgebung suchen können.

Kapitel 1 – Einstieg

In diesem Kapitel werden wir zum einen auf den allgemeinen Aufbau von Rails und zum anderen auf die geschichtliche Entwicklung des Frameworks eingehen. Des Weiteren werden wir in diesem Kapitel eine minimale Rails-Applikation erstellen, die bereits mit der Datenbank kommuniziert. Anhand dieser kleinen Beispielapplikation wollen wir die einzelnen Komponenten von Rails kurz anreißen und dann in den nachfolgenden Kapiteln konkreter erläutern. Das erste Kapitel soll uns somit ein vollständiges Bild von Rails vermitteln, ohne auf spezielle Details der einzelnen Komponenten einzugehen.

Kapitel 2 – ORM-Bibliotheken (Model-Komponente)

Die Model-Komponente stellt das „M“ im MVC-Framework dar. Innerhalb dieses Kapitels werden wir auf das Thema ORM-Bibliothek eingehen sowie seine Einsatz- und Verwendungsmöglichkeiten erläutern. In Ruby gibt es eine Vielzahl von ORM-Bibliotheken wie beispielsweise Sequel, DataMapper und ActiveRecord. Wir werden in diesem Kapitel detaillierter auf die ORM-Bibliothek DataMapper eingehen. Die übrigen Beispiele im Buch verwenden die ORM-Bibliothek ActiveRecord.

Kapitel 3 – Template (View-Komponente)

Die View-Komponente stellt das „V“ im MVC-Framework dar. In diesem Kapitel gehen wir primär auf die Template-Sprache Haml für die Generierung von HTML und auf die Metasprache Sass für das Generieren von CSS ein.

Kapitel 4 – Programmlogik (Controller-Komponente)

Neben dem „M“ und dem „V“ im MVC-Framework fehlt uns nun nur noch das „C“, die Controller-Komponente. In diesem Kapitel gehen wir darauf ein, wie eine HTTP-Anfrage von der Rails-Applikation konkret verarbeitet wird. Auch weiterführende Themen wie die Ruby-Webserver-Schnittstelle Rack und der Softwarearchitekturstil „Representational State Transfer“, der unter der Abkürzung REST bekannt ist, finden in diesem Kapitel ihre Berücksichtigung.

Kapitel 5 – E-Mail-Nachrichten verarbeiten (ActionMailer)

Mit dem ActionMailer können wir E-Mail-Nachrichten unter Rails sowohl versenden als auch empfangen. In Rails 3 wurde diese Komponente vollständig überarbeitet und die Handhabung deutlich vereinfacht.

Kapitel 6 – Testen

Im Kapitel „Testen“ gehen wir auf die Grundaspekte des Schreibens von automatisierten Tests ein. Darüber hinaus stehen aber auch Themen wie die Nutzung und die Installation eines Continuous Integration Servers (kurz CI-Server) innerhalb dieses Kapitels im Vordergrund.

Kapitel 7 – Internationalisierung und Lokalisierung

Beim Thema Internationalisierung und Lokalisierung handelt es sich um ein sehr umfangreiches Themengebiet. Wir werden in diesem Kapitel zeigen, welche Möglichkeiten Rails standardmäßig bietet, um Applikationen, die mit dem Framework entwickelt wurden, zu internationalisieren bzw. zu lokalisieren.

Kapitel 8 – Werkzeugkasten

In diesem Kapitel zeigen wir, wie Sie konkrete Herausforderungen aus der Softwarewelt am besten lösen können. Themen wie das Erstellen einer einfachen Blätterfunktion oder eines PDF-Dokuments, aber auch Themen wie das Durchsuchen von großen Datenmengen mithilfe einer Full-Text Search Engine werden in diesem Kapitel näher erläutert. Alle Leser, die bereits Erfahrung mit Ruby und dem Framework Rails haben, werden in diesem Kapitel sicherlich noch einige gute Anregungen bzw. Tipps finden.

1.4 Voraussetzungen

Dieses Buch erläutert die einzelnen Themengebiete anhand von zahlreichen praxisorientierten Beispielen. Für ein besseres Verständnis empfehlen wir Ihnen, einzelne Passagen abzutippen und dabei auch gern einen Schritt weiter zu gehen als im Buch, frei nach dem Motto „learning by doing“. Selbstverständlich werden ausgewählte Quelltextbeispiele auf der Webseite zum Buch zu Ihrer Verfügung bereitgestellt. Sie benötigen einen Computer, auf dem die Programmiersprache Ruby sowie der zugehörige Paketmanager RubyGems installiert sind. In diesem Buch sind die meisten Beispiele mit einer MySQL-5-Datenbank im Hintergrund umgesetzt. Hierbei ist es irrelevant, ob Sie als Betriebssystem

Windows, Linux oder Mac OS X verwenden. Die Programmiersprache Ruby ist betriebssystemunabhängig. Wir persönlich bevorzugen Mac OS X, was an einigen Stellen des Buches durchaus sichtbar sein kann.

Bei konkreten Fragen können Sie uns jederzeit eine E-Mail schreiben. Gegebenenfalls werden wir Fragen zusammenfassend auf der Webseite zum Buch beantworten.

1.5 Ruby on Rails

Bis jetzt ist der Begriff „Rails“ schon ein dutzend Mal gefallen. Aber was ist Rails eigentlich genau? Bei Rails handelt es sich um ein Open-Source-Webframework, welches vom dänischen Programmierer David Heinemeier Hansson entwickelt wurde. Er extrahierte Rails aus seiner Arbeit am Projektmanagementtool Basecamp und präsentierte es Mitte 2004 das erste Mal der Öffentlichkeit. Das Webframework Rails basiert auf der „Model View Controller“-Architektur (kurz MVC). Die MVC-Architektur unterteilt die eigene Softwarestruktur in eine Model- (Model), Präsentations- (View) und Steuerungsschicht (Controller). Wie der Name bereits erkennen lässt, ist Rails in der Programmiersprache Ruby geschrieben. Im Laufe der Entwicklung von Rails ist ein modulares Webframework entstanden. So legt uns Rails nicht auf eine bestimmte Template-Sprache (z. B. ERB, Haml), JavaScript-Bibliothek (z. B. jQuery, Prototype) oder einen bestimmten OR-Mapper (z. B. ActiveRecord, Sequel, DataMapper) fest.

Rails unterliegt dem Prinzip „Convention over Configuration“, das heißt, dass Rails gewisse Konventionen vorgibt. Beispielsweise entspricht der Tabellenname in der Datenbank dem Plural des Modelnamens, wenn wir den Tabellennamen im Model nicht explizit über die Methode *set_table_name* setzen. In verschiedenen anderen Frameworks zur Erstellung von Software ist es üblich, zahlreiche Dinge in Konfigurationsdateien vorkonfigurieren zu müssen, bevor man erste Ergebnisse sehen konnte. Dies ist bei Rails nicht der Fall, vorausgesetzt, wir folgen den Konventionen, die Rails vorgibt. Hierdurch sind eine beschleunigte Softwareentwicklung und rasche Umsetzung von Ideen möglich.

Des Weiteren wurde die Webentwicklung von Rails durch den Leitsatz „Don’t Repeat Yourself“ (DRY) geprägt. Das bedeutet, dass der Quelltext einer bestimmten Funktionalität nicht in mehrere Methoden dupliziert wird und somit Wiederholungen vermieden werden. Rails prägte die Welt der Webentwicklung ebenfalls im Bereich des Testens von Applikationen, da Rails von Haus aus ein gut integriertes und somit bequem nutzbares Testframework beinhaltet. Das gibt dem Entwickler die Möglichkeit, den Test noch vor der eigentlichen Implementierung zu schreiben. Dieser Entwicklungstil wird auch als „Test-driven Development“ (kurz TDD) bezeichnet.

1.5.1 Historische Entwicklung

Rails hat sich von 2005 bis zum heutigen Tag enorm weiterentwickelt. Lassen Sie uns nun kurz die genaue Entwicklung von Rails nachvollziehen.

Version 1.0

Im Dezember 2005 wurde die Version 1.0 fertiggestellt. Ziel dieser Version war es, Rails nach seiner Extraktion aus Basecamp auf ein solides Grundfundament zu setzen, sodass Webanwendungen für den produktiven Einsatz mithilfe von Rails entwickelt werden konnten.

Version 1.1

Bereits vier Monate später, im März 2006, wurde die Version 1.1 veröffentlicht. Neben den mehr als 500 Bugfixes wurden auch neue Funktionen wie RJS-Templates hinzugefügt. RJS-Templates erzeugen JavaScript-Quelltext, der ausgeführt wird, sobald dieser an den Browser zurückgegeben wird. Weiterhin wurden beim OR-Mapper ActiveRecord einige Veränderungen vorgenommen. Es war nun möglich, M:N-Beziehungen über eine konkrete Model-Klasse zu modellieren (*has_many :friends, :through => :friendships*). Auch die polymorphe Beziehung wurde neu eingeführt. Hiermit können wir ausdrücken, dass die Model-Klassen User und Event dasselbe Address-Model verwenden, sodass es in der Datenbank nur eine Tabelle für die Adressen gibt. Für eine bessere Performance bei der Nutzung von ActiveRecord wurde das JOIN-ing zwischen Model-Klassen optimiert. Über die Funktionalität von *respond_to* war es jetzt möglich, im Controller besser auf Anfragen mit verschiedenen Content-Typen zu reagieren und somit unterschiedliche Formate als Antwort zurückzuliefern.

Version 1.2

Ein dreiviertel Jahr später, im Januar 2007, erschien die Version 1.2. Die Hauptneuerung in dieser Version war die Integration des Softwarearchitekturstils REST (Representational StateTransfer). Auf diese Technologie werden wir im Kapitel „Programmlogik (Controller-Komponente)“ genauer eingehen. Darüber hinaus wurden zahlreiche Methoden des Frameworks als deprecated (dt. veraltet) gekennzeichnet, die in Rails 2.0 endgültig gelöscht wurden. Durch die starke Verbreitung von Rails über den englischsprachigen Raum hinaus, wurde auch die Behandlung von UTF-8-Zeichenketten verbessert.

Version 2.0

Knapp ein Jahr Entwicklungszeit ging ins Land, bis die Version 2.0 im Dezember 2007 fertig gestellt wurde. Das Major Release 2.0 von Rails brachte weit über einhundert Neuerungen und Verbesserungen mit sich. Neben der Migration von SOAP zu REST wurden auch weitere Funktionen wie die Authentifizierung über HTTP-Basic-Authentication integriert. Die Syntax für die Schreibweise von Migrationen wurde mehr in Richtung DRY verändert (Sexy migrations). Das Referenzieren zwischen verschiedenen Fixtures wurde ebenfalls deutlich vereinfacht (Foxy fixtures).

Version 2.1

Rails 2.1 erschien im Juni 2008. Neuerung in dieser Version war unter anderem die verbesserte Berücksichtigung verschiedener Zeitzonen. Über die Funktionalität „dirty object“ von ActiveRecord hat man vor dem Abspeichern eines Model-Objektes die Möglichkeit, Veränderungen des Objekts anzeigen zu lassen. Mithilfe der Deklaration (*config.gem("haml", :version => "2.2.15")*) in der Datei *config/environment.rb* können wir die Abhängigkeit von der Rails-Applikation zu den verwendeten GEM-Bibliotheken definieren. Häufig verwendete ActiveRecord-Finder-Definitionen können wir nun in *named_scope*-Deklarationen der Model-Klasse kapseln. Eine weitere große Verbesserung war die Umstellung der Versionsnummern der Datenbankmigrationen auf Timestamps, da es zuvor beim Arbeiten im größeren Team immer zu Versionskonflikten gekommen war. Des Weiteren wurden noch einige Optimierungen bei der Caching-Implementierung vorgenommen.

Version 2.2

Im November 2008 wurde die Rails-2.2-Version freigegeben. Seit dieser Version ist Rails mit Ruby 1.9.x und JRuby kompatibel. Die Hauptneuerung in dieser Version war die Integration der I18n-Gem-Bibliothek. Seitdem stellt Rails ein Standard-API für die Internationalisierung bereit. Für das Cachen auf HTTP-Ebene unterstützt Rails 2.2 etag und last-modified im HTTP-Header. Die Thread-Sicherheit von Rails wurde ebenfalls verbessert und ActiveRecord bekam einen „database connection pool“.

Version 2.3

Rails 2.3 erschien im März 2009. Der Umgang mit verschachtelten Model-Klassen wurde dank „nested forms“ stark vereinfacht. Die wohl größte und bedeutendste Änderung dieser Version war die Integration der Webserverschnittstelle Rack. Weiterhin wurde das Modul Rails Metal eingeführt. Mithilfe von Rails Metal ist es möglich, Requests zu beantworten, bevor sie durch den gesamten Stack von Rails gelaufen sind. Dies verringert die Antwortzeit eines HTTP-Requestes deutlich.

Version 3.0

Begonnen hat die Planung für die Version 3.0 bereits im Dezember 2008, als David Heinemeier Hansson am 23. Dezember 2008 ankündigte, dass die besten Funktionen von Rails und dem anderen großen Ruby-Webframework Merb (siehe nächster Abschnitt) zu Rails 3.0 zusammengefügt werden sollen. Hauptkritikpunkt an Rails war bis zu dieser Zeit, dass es nicht gut skaliert und nicht ausreichend modular aufgebaut war. Fest steht, dass Rails 3.0 erst durch den Einfluss von Merb an Modularität und Performance gewonnen hat. Als größte Neuerungen in dieser Version sind die Möglichkeiten der freien Wahl des OR-Mappers (z. B. ActiveRecord, Sequel, DataMapper), der Template-Sprache (z. B. ERB, Haml) und der JavaScript-Bibliothek (z. B. jQuery, Prototype) zu nennen. Weitere neue Funktionalitäten in Rails 3.0 sind unter anderem der Bundler für Gem-Bibliotheken, der neue Router und die Arel-Syntax bei ActiveRecord.

HINWEIS: Merb ist wie Rails ein MVC-Web Framework, welches ebenfalls in Ruby geschrieben ist. Die Geschichte von Merb begann am 22. September 2006, als Ezra Zygmuntowicz auf pastie.org den ersten Quelltext veröffentlichte (siehe Anhang A.1). Er hatte das Problem, dass seine Rails-Applikation blockiert war, während größere Dateien hochgeladen wurden. Dies wollte er mithilfe eines kleinen Frameworks lösen. Der Name Merb entstand aus der Kombination Mongrel, den er als HTTP-Server nutzte, und ERB, welches er als Template-System einsetzte. Er war natürlich nicht der Einzige, der zuvor beschriebenes Problem hatte, sodass auch viele andere Rails-Nutzer bemerkten, dass Rails für einige Anwendungsfälle zu schwergewichtig war. Somit entstand im Laufe der Jahre ein Webframework, bei dem das Augenmerk stark auf Performance und Skalierbarkeit liegt. Weitere Informationen finden Sie auf der offiziellen Webseite von Merb (*http://www.merbivore.com*).

1.5.2 Die verschiedenen Infoquellen

Rails

- Offizielle Webseite: *http://www.rubyonrails.org*
- Offizieller Blog: *http://weblog.rubyonrails.org*
- Was ist neu in Rails (Edge): *http://edgerails.info*
- Wiki: *http://wiki.rubyonrails.org*
- Rails-Guide: *http://guides.rubyonrails.org*
- IRC: irc.freenode.net#rubyonrails
- API-Dokumentation: *http://api.rubyonrails.org*
- Metablog: *http://planetrubyonrails.com*
- Mailingliste (EN): *http://groups.google.com/group/rubyonrails-talk*
- Mailingliste (DE): *http://www.rubyonrails-ug.de*

Ruby

- Offizielle Ruby-Webseite: *http://www.ruby-lang.org*
- Dokumentation: *http://ruby-doc.org*

1.5.3 Komponenten

Den kompletten Funktionsumfang der einzelnen Komponenten können Sie ebenfalls in der Rails-API-Dokumentation nachlesen.

ActionPack

Die Komponente ActionPack hat in Rails eine zentrale Aufgabe, da sie den Request über den Router annimmt und ihn auf den dazugehörigen Controller und eine Action weiterleitet. Nach der Verarbeitung des Requests durch den Controller gibt ActionPack die Antwort an den Client zurück. Dazu spricht ActionPack die benutzte Template Engine (z. B. Haml, ERB) an und gibt den Inhalt der fertig gerenderten Templates an den Client zurück. Die Komponente ActionPack implementiert somit sowohl die Bestandteile für die Controller-Schicht als auch für die View-Schicht der MVC-Architektur.

ActiveRecord

ActiveRecord stellt in der MVC-Architektur das „M“ (Model) dar. Es ist wie DataMapper oder Sequel ein OR-Mapper und bietet uns eine Abstraktionsschicht zur Datenbank. Das Subframework ActiveRecord ist nach einem Entwurfsmuster (engl. Design Pattern) benannt, welches Martin Fowler in seinem Buch „Patterns of Enterprise Application Architecture“ genau erläutert.

ActiveResource

In Rails 2.0 wurde ActionWebService durch die Komponente ActiveResource ersetzt. Mithilfe von ActiveResource können REST-konforme Webservices (Representational State-Transfer) in Applikationen eingebaut und konsumiert werden.

ActiveModel

Mithilfe der Komponente ActiveModel möchte man vermeiden, dass die Entwicklung einzelner Model-Funktionalitäten wie Observer, Callbacks, Validations oder Serialization auseinanderlaufen. Durch Extraktion von ActiveModel ist es möglich, diese Funktionalität identisch für ActiveRecord und ActiveResource bereitzustellen. Weiterhin können wir diese herausgelösten Funktionen auch in eigene Klassen integrieren.

ActiveSupport

ActiveSupport ist eine Sammlung von nützlichen Klassen und Erweiterungen der Standard-Ruby-Bibliothek, mit denen sich Standard-Programmieraufgaben eleganter lösen lassen. Um die hilfreichen Erweiterungen auch in anderen Bibliotheken nutzen zu können, wurden diese in eine eigene Bibliothek gekapselt. Zahlreiche Subframeworks (z. B. ActiveRecord, ActiveResource) von Rails weisen deshalb eine Abhängigkeit zu dieser Bibliothek auf. ActiveSupport stellt uns beispielsweise zahlreiche Datumshilfsfunktionen bereit. So liefert z. B. die Instanzmethode *end_of_month* eines Date-Objekts den letzten Tag des Monats zurück (*Date.today.end_of_month => Mon, 31 May 2010*).

ActionMailer

ActionMailer ist ein Subframework von Rails, das uns einen abstrahierten Layer für das Senden, Empfangen und Verarbeiten von E-Mails bereitstellt.

Railties

Railties verbindet die vorgenannten Komponenten zu einem Webframework, das wir insgesamt als Rails bezeichnen. Die Komponente Railties umfasst neben den zahlreichen Generatoren für das Anlegen von Controllern, Migrationen, Model-Klassen und dem Generieren von neuen Rails-Applikationen auch das Konfigurationsmanagement der einzelnen Komponenten.

1.6 Lernen am Beispiel

Ganz nach dem Leitsatz „Learning by doing" wollen wir Ihnen die Funktionsweise von Rails anhand einer sehr einfachen Applikation vermitteln, sodass Sie einen Gesamtüberblick über das Webframework Rails erhalten. Die Rails-Applikation Kraeftemessen.com (Veranstaltungsdatenbank) werden wir erst in den Folgekapiteln als Beispiel für die jeweiligen Programmierthematiken heranziehen. Für die erste Rails-Applikation haben wir uns überlegt, eine einfache Einkaufsliste (engl. shopping list) zu entwickeln, bei der wir Positionen hinzufügen, aktualisieren und löschen können. Wir denken, dass dies eine sehr einfache und klare Aufgabenstellung ist, die wir nun gemeinsam umsetzen wollen.

1.6.1 Installation

Da es schwierig ist, eine allgemeingültige Installationsanleitung zu verfassen, möchten wir Ihnen an dieser Stelle lediglich die wesentlichen Anhaltspunkte für die Installation Ihrer lokalen Entwicklungsumgebung geben. Kommen Sie an einem bestimmten Punkt der Installation nicht weiter oder haben dabei ein Problem, dann können Sie uns natürlich jederzeit kontaktieren. Gegebenenfalls werden wir erweiterte Informationen auf der Webseite zum Buch bereitstellen. Oder Sie verwenden eine von zahlreichen im Internet verfügbaren Installationsanleitungen. Befragen Sie hierfür einfach eine Suchmaschine wie bing.de oder google.de (z. B. „installation rails windows").

Microsoft Windows

Die Installation von Rails und dessen Komponenten unter Microsoft Windows unterscheidet sich deutlich von anderen Betriebssystemen. Zuerst müssen wir Ruby und den dazugehörigen Paketmanager RubyGems installieren. Hierfür steht uns ein One-Click Installer zur Verfügung, den wir von der Webseite *http://www.ruby-lang.org/de/downloads* herunterladen können. Das Installationsskript installiert Ruby standardmäßig im Verzeichnis *c:\ruby*. Vergewissern Sie sich, dass sich der Pfad *c:\ruby\bin* im Suchpfad PATH Ihrer Umgebung befindet, ansonsten können Sie die nachfolgenden Befehle nicht auf der Kommandozeile ausführen. Sie können dies über den Befehl *path* auf der Kommandozeile überprüfen.

Lassen Sie uns als Erstes den Paketmanager RubyGems aktualisieren.

```
01| $> gem update --system
```

Anschließend installieren wir das Webframework Rails über den Paketmanager RubyGems

```
01| $> gem install rails --pre
```

Wenn Sie nicht bereits einen MySQL-Server auf Ihrem Computer installiert haben, laden Sie sich bitte das Installationsskript von der MySQL-Webseite *http://dev.mysql.com/downloads/mysql/5.1.html#win32* herunter und führen danach das Installationsskript aus.

Nachdem wir den MySQL-Server installiert haben, benötigen wir die Ruby-Bibliothek für die Kommunikation mit der MySQL-Datenbank, die wir ebenfalls über den Paketmanager RubyGems installieren können.

```
01| $> gem install mysql
```

Sie sollten nun eine funktionsfähige Entwicklungsumgebung für Rails auf Ihrem Computer installiert haben.

Mac OS X

Mac OSX Leopard beinhaltet Rails bereits in der Version 1.2.6 und Ruby in der Version 1.8.6. Rails 3.0 setzt Ruby 1.8.7 oder eine aktuellere Version voraus. Wer bereits Mac OS X in der Version Snow Leopard installiert hat, kann die folgende Installation von Ruby überspringen, da Ruby 1.8.7 bereits installiert ist.

Im Kapitel „Werkzeugkasten" erläutern wir, wie wir mehrere Ruby-Versionen über die Gem-Bibliothek RVM (Ruby Version Manager) verwalten können. Wir können Ruby aber auch über den Paketmanager MacPorts (*http://www.macports.org*) installieren, oder Ruby wie folgt selber kompilieren.

```
01| $> wget ftp://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.1-p376.tar.gz
02| $> tar xzvf ruby-1.9.1-p376.tar.gz
03| $> cd ruby-1.9.1-p376
04| $> ./configure
05| $> make
06| $> sudo make install
```

- **Zeile 01:** Sollten Sie *wget* nicht installiert haben, können Sie sich den Quelltext von der Webseite *http://www.ruby-lang.org/de/downloads/* herunterladen.

Der Paketmanager RubyGems wird wie folgt aktualisiert:

```
01| $> sudo gem update --system
```

Die aktuelle Version von Rails installieren wir über den Paketmanager RubyGems.

```
01| sudo gem install rails --pre
```

Wenn Sie nicht bereits einen MySQL-Server auf Ihren Rechner installiert haben, laden Sie sich bitte das Installationsskript von der MySQL-Webseite *http://dev.mysql.com/downloads/mysql/5.1.html#macosx-dmg* herunter und führen danach das Installationsskript aus.

Nachdem wir den MySQL-Server installiert haben, benötigen wir die Ruby-Bibliothek mysql für die Kommunikation mit der MySQL-Datenbank, die wir ebenfalls über den Paketmanager RubyGems installieren können.

```
01| $> sudo env ARCHFLAGS="-arch x86_64" gem install mysql -- \
02|         --with-mysql-dir=/usr/local/mysql \
03|         --with-mysql-lib=/usr/local/mysql/lib \
04|         --with-mysql-include=/usr/local/mysql/include
```

Die verwendeten Pfade können bei Ihrer MySQL-Installation abweichen. Abhängig von der Art der Installation (32 oder 64 bit), variiert auch der Wert des Parameters „ARCHFLAGS“.

Sie sollten nun eine funktionsfähige Entwicklungsumgebung für Rails auf Ihrem Computer installiert haben.

Linux

Die Installationsschritte hängen stark von der gewählten Distribution (z. B. Ubuntu, Debian, CentOS) ab. Am bequemsten ist die Installation der Komponenten über den Paketmanager der gewählten Distribution.

Finden wir keine fertigen Installationspakete, haben wir die Möglichkeit, Ruby selber zu kompilieren. Die aktuelle Ruby-Version finden wir auf der Webseite *http://www.ruby-lang.org/de/downloads/* zum Herunterladen.

```
01| $> wget ftp://ftp.ruby-lang.org/pub/ruby/1.9/ruby-1.9.1-p376.tar.gz
02| $> tar xzvf ruby-1.9.1-p376.tar.gz
03| $> cd ruby-1.9.1-p376
04| $> ./configure
05| $> make
06| $> sudo make install
```

Ruby in der Version 1.9.1 beinhaltet bereits den Paketmanager RubyGems, sodass wir Rails direkt über den Paketmanager installieren können.

```
01| $> sudo gem install rails --pre
```

Wenn Sie nicht bereits einen MySQL-Server auf Ihren Rechner installiert haben, stehen auf der MySQL-Webseite *http://dev.mysql.com/downloads/mysql/5.1.html* für einige Distributionen fertige RPM-Pakete bereit.

Nach der Installation des MySQL-Servers benötigen wir die Ruby-Bibliothek mysql für die Kommunikation mit der MySQL-Datenbank, die wir ebenfalls über den Paketmanager RubyGems installieren können.

```
01| $> gem install mysql
```

Sie sollten nun eine funktionsfähige Entwicklungsumgebung für Rails auf Ihrem Computer installiert haben.

1.6.2 Eine Rails-Applikation erstellen

Lassen Sie uns nach der erfolgreichen Installation von MySQL, Ruby, RubyGems und Rails damit beginnen, eine erste Rails-Applikation zu generieren. Hierfür stellt uns Rails das Kommandozeilentool *rails* bereit. Über die Option *help* bekommen wir eine kurze Dokumentation der jeweiligen Optionen angezeigt (*rails --help*). Standardmäßig wird eine Rails-Applikation mit der Konfiguration für eine SQLite-Datenbank generiert. Beim Erstellen der Rails-Applikation können wir über die Option *database* angeben, welche Datenbank wir verwenden möchten. Zur Auswahl steht uns neben dem Standardwert *sqlite3* auch *mysql*, *oracle*, *postgresql*, *frontbase* und *ibm_db*. Lassen Sie uns nun wie folgt unsere erste Rails-Applikation erstellen:

```
01| $> rails new shopping_list --database=mysql
02|       create
03|       create  README
04|       create  Rakefile
05|       create  config.ru
06|       create  Gemfile
07|       create  app
08|       create  app/controllers/application_controller.rb
09|       ...
```

Anschließend befindet sich im aktuellen Arbeitsverzeichnis ein neues Verzeichnis *shopping_list* mit der neuen Rails-Applikation.

1.6.3 Verzeichnisstruktur von Rails

Rails erzeugt uns beim Erstellen einer neuen Rails-Applikation eine Vielzahl von Unterverzeichnissen. In der folgenden Tabelle wollen wir Ihnen die Bedeutung der einzelnen Verzeichnisse erläutern.

Verzeichnis	Beschreibung
app	Stellt das Herzstück der Applikation dar. In diesem Verzeichnis befinden sich standardmäßig die Controller, Helper, Model-Klassen und Views der Applikation.
app/controllers	In diesem Verzeichnis liegen die Controller der Applikation, die den Programmfluss der Applikation steuern.
app/helpers	Hier haben wir die Möglichkeit, Helper-Methoden abzulegen, die wir dann in den Views verwenden können.