Editor: Christof Ebert ■ Vector Consulting ■ christof.ebert@vector-consulting.de

# Ruby on Rails

**Michael Bächle and Paul Kirchberg**

*Web applications are getting ever more complex, and many of us wonder how to master the technology challenges of creating highly dynamic, good-looking pages without too much overhead. Ruby on Rails is an open source framework developed to increase programmer productivity and reduce entry barriers to programming Web applications. In this issue's column, Michael Bächle and Paul Kirchberg look into RoR and contrast it with other frameworks.*

*I look forward to hearing from both readers and prospective column authors, about this column and the technologies you want to know more about. —Christof Ebert*

Web 2.0, Ajax (Asynchronous JavaScript and XML), Web services, and related technologies are complicating Web application engineering. As application complexity increases, so does the wish for a silver bullet. A lot of Web 2.0 frameworks promise to make developers' lives easier. But do they?

Ruby on Rails is a novel Web 2.0 framework that attempts to combine PHP's simple immediacy with Java's architecture, purity, and quality. It's certainly one of the most promising frameworks that have emerged for agile Web 2.0 application development. Here, we'll first look briefly at underlying RoR principles and then take a closer look at its components.

## RoR principles

RoR forms an environment and provides all the tools for developing business-critical, database-supported Web applications. It includes all the so-called Web 2.0 features, such as support for Ajax. Its basic objectives, however, are simplicity, reusability, expandability, testability, productivity, and maintainability. Six RoR principles contribute significantly to these objectives.

### MVC architecture

RoR implements a *model-view-controller* architecture. MVC clearly separates code according to its purpose. Three subframeworks play a significant part in this separation: Active Record, Action View, and Action Controller. These subframeworks appear across the middle of the component diagram in figure 1.

### Convention over configuration

RoR avoids configurations in their broadest sense. The only configuration file is called `database.yml`; it contains the type, user name, and password for an application database. For every project, developers can define three different databases: `development`, `test`, and `production`. The RoR framework firmly integrates the RoR components to cooperate even without configuration. Developers can overwrite conventions to make special functions and individual adjustments possible.

0740-7459/07/$25.00 © 2007 IEEE

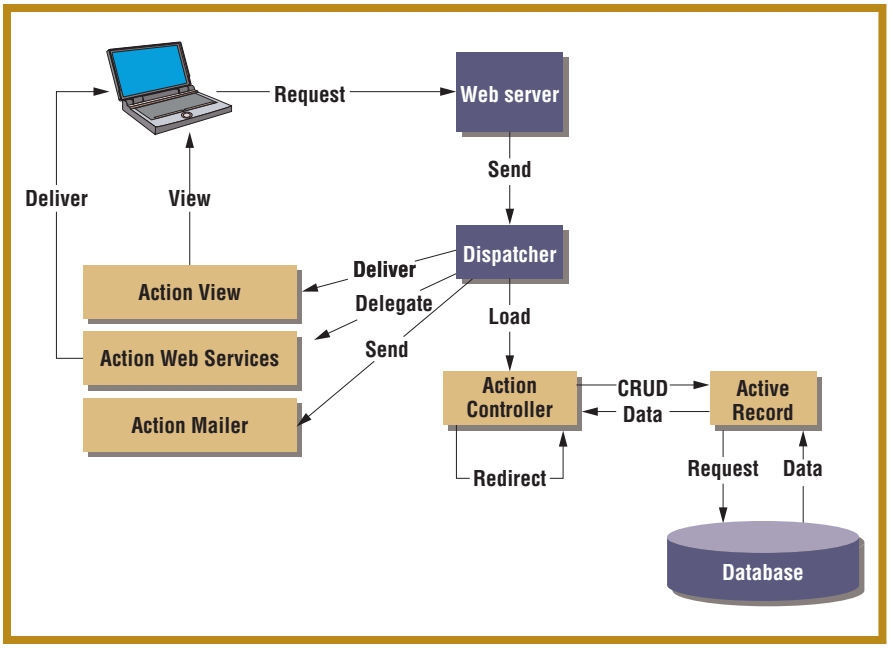


**Figure 1. Ruby on Rails framework components. The Active Record, Action View, and Action Controller subframeworks establish a fundamental model-view-controller architecture.**

```
File A: ../app/controllers/hello_controller.rb

class HelloController < ApplicationController
  def index
    @message = "Hello World!"
  end
end

File B: ../views/hello/index.rhtml

<html>
  <body>
    <%= @message %>
  </body>
</html>
```

**Figure 2. The "Hello World" program in Ruby. File A is the Ruby code listing; File B is the HTML template for hello_controller.rb.**

### Don't Repeat Yourself

The DRY principle requires every piece of system knowledge to have a unique, definite, and relevant representation. This approach means developers repeat themselves seldom or never. In this context, a piece of knowledge can be data or metadata, logic, functionality, or an algorithm. As an environment, RoR is designed to make singular declarations and reuse as easy as possible. The DRY principle is striking, for example, in Active Record, which determines attributes and class values exclusively in the database and not in the program code.

### Scaffolding

The database scheme and the `scaffold` console command let RoR create a basic skeleton of controllers and view templates that have create-retrieve-update-delete (CRUD) functionality. In many programming situations, RoR also offers generators that save a lot of time for recurring elements, such as a login form.

The automatic creation of project scaffolding saves time and lets the developer immediately start work on the application's core functionality.

### Immediate feedback

Developers can get an up-to-date status of their program by reloading it in the browser. No additional time is involved in deploying the program. RoR doesn't require projects to compile and execute first.

### Ruby

RoR is based on the dynamically typed, object-oriented Ruby programming language. Yukihiro Matsumoto introduced this language in 1995. Immediately when the browser calls up Ruby code, the Web server's Ruby interpreter processes it. In comparatively little code, developers can cover a large amount of functionality by combining loops and arrays in blocks. Ruby code is easy to read and based heavily on the languages Perl, Python, Smalltalk, and Lisp.

## RoR framework components

Figure 1 shows the most important RoR framework components. Figure 2 presents a programming example in Ruby, the inevitable "Hello World" program.

### Active Record

The Active Record subframework establishes the connection between the domain objects and the database. It transforms the CRUD functions that come from the Action Controller into SQL commands, sends the requests to the database, and returns the received results to the Action Controller. Active Record also validates whether a user is entitled to access or change a specific record.

The subframework follows the Object/Relational Mapping pattern. In ORM, classes are directly related to database tables. The records represent class objects; the table columns represent the attributes. Usually, ORM requires a comprehensive configuration via XML. The Active Record conventions enable

RoR to avoid this requirement. Via `ActiveRecord::Base`, RoR activates tables when they're called in the database and creates the required class automatically. The "convention over configuration" principle bypasses the configuration effort usually required to combine data and variables. Active Record classes thus receive their attributes directly from the database table definition.

To represent inheritance in a relational database, RoR uses an additional pattern, Single Table Inheritance. RoR associations are defined by a foreign key in the database and a declaration of the corresponding table name (in plural by convention, but the developer can change this) in Active Record. A collection of validation methods lets RoR automatically check entered data in the HTML form regarding its existence or specific format. If the validation fails, RoR creates an error message and displays it in the HTML form (we describe this further in "Action View").

### Action Controller

The Action Controller is the central steering unit. It receives requests and returns a view to the client. Within this component, the developer defines *actions* that determine how the controller will react to specific HTTP requests. The controller receives a request from a URL, processes it as an action or object and sends the right view back to the browser. RoR uses Pretty URLs, which the Apache Web server, for example, makes possible through the mod_rewrite function.

Additionally, the Action Controller can run an action through a before-filter prior to performing it and through an after-filter upon completion. The before-filter can log specific actions, such as forwarding to a login page. The after-filter is especially useful in handling options, such as checking whether the user wants an encoded transfer method for the server's response.

Action Controller also establishes the connection to Active Record and supplies data from the database to Action View, Web Services, and Mailer. Additional functions include, for example, session handling.

### Action View

Action View presents data. Typically, RoR represents data in the HTML format. In any case, RoR defines templates to separate the presentation layer from the rest of the application. RoR has various template forms. The most important are RHTML and RXML.

RHTML templates mix Ruby code with HTML; during runtime, processing the Ruby code fills the templates with content. In this case, the RHTML template encloses the Ruby code in a tag of the form `<%...%>`.

To create XML files, RoR uses its own builder library. The builder transfers Ruby commands into XML tags. RoR makes preimplemented helper methods available. These methods are especially useful for format conversions, format boxes, error messages, and site management.

### Action Web Services

Through Action Web Services, RoR supports XML-RPC-based and SOAP-based Web services, thus facilitating the publishing of functionality and the inclusion of external services. Active Controller supports the lighter-weight REST (*Re*presentational *S*tate *T*ransfer) approach.

RoR also supports Ajax frameworks such as Prototype and script.aculo.us, which makes it easier for developers to use Ajax to design interactive, user-friendly RoR applications.

**In Ruby on Rails, Active Record classes receive their attributes directly from the database table definition.**

## Alternative Web frameworks

There are many alternative Web 2.0 frameworks. The four we introduce here are among the most popular.

### Apache Struts

This Java-based Web framework is a classic representative of the MVC architecture. Struts focuses on the interaction between the controller and view. The controller component is a Java servlet that defines process control in a configurable XML file. JavaBeans provides access to the model's data, and JavaServer pages represent the view component.

For more information, see http://struts.apache.org.

### Tapestry

In contrast to the action-driven Struts, Tapestry follows an event-oriented approach that also lets developers implement Web applications according to MVC pattern. Because the Tapestry approach is event-oriented, it corresponds more to the implementation of user interfaces with Swing or the Abstract Window Toolkit—that is, it strongly abstracts from the actual technology of the Web.

For more information, see http://tapestry.apache.org.

### Cocoon

As an XML publishing framework, Cocoon follows a completely different approach. It focuses on representing data that either already exists in XML format or will be stored in XML format. Cocoon sets up a flexible presentation layer on this XML database to support separation between presentation and content. It bases document publishing on XSLT (Extensible Style Sheet Language Transformation).

For more information, see http://cocoon.apache.org.

### ASP.NET

Comparable to Java 2 Enterprise Edition, Microsoft's .NET framework comprises an application platform that also supports distributed server-based applications. ASP.NET is the central component for Web development. It lets developers choose their programming

## Ruby on Rails Resources

Probably the standard book for Ruby on Rails is *Agile Web Development with Rails* by Dave Thomas and David Heinemeier Hansson (2nd ed., Pragmatic Bookshelf, 2007).

For a brief overview of RoR and what Java can learn from it, see Bruce Tate's "Crossing Borders: What's the Secret Sauce in Ruby on Rails?" 2006; www-128.ibm.com/developerworks/java/library/j-cb05096.html.

Aaron Rustad compares two frameworks in "Ruby on Rails and J2EE: Is There Room for Both?" 2006; www-128.ibm.com/developerworks/web/library/wa-rubyonrails.

The official RoR site is www.rubyonrails.org. For a list of Ruby projects, see http://rubyforge.org.

language from all available .NET languages. One advantage of the ASP.NET approach is the combination of client- and server-based programming. So, for example, developers can use the same script languages for both and can determine the processing place via an option.

For more information, see www.asp.net.

We used RoR to develop an alumni system (http://ali.ba-ravensburg.de and http://rubyforge.org/projects/alumnionrails) on a Solaris operating system with a MySQL database management system and Apache server. RoR proved a stable, sophisticated, state-of-the-art framework for agile Web engineering. Although we've briefly discussed only its main components here, RoR has other features to support efficient Web engineering, such as test support. The "Ruby on Rails Resources" sidebar has more information.

RoR embodies the basic principles of the Agile Manifesto (http://agilemanifesto.org). It abstains from heavy tools and focuses instead on "individuals and their interactions." The principles of scaffolding and immediate feedback provide a first operable version early in the development cycle. The system is developed incrementally in small steps. The scaffolding gives developers an operable version of the current system at all times. This helps developers work efficiently in cooperation with the customer, discuss all topics directly in relation to the running system, and implement changes much faster. However, as in all Web frameworks, good software is still the work of good professional developers. Like all frameworks, RoR is no "silver bullet."

**Michael Bächle** is a professor of business information systems at the University of Cooperative Education, Ravensburg, Germany. Contact him at baechle@ba-ravensburg.de.

**Paul Kirchberg** is a professor of business information systems at the University of Cooperative Education, Ravensburg, Germany. Contact him at kirchberg@ba-ravensburg.de.