CMPUT 410

Ruby on Rails 3

March 14th, 2011

Ken Li
Allison Pon
Kyra Leimert
Matt Delaney
Edward Bassett

Introduction - What is Ruby on Rails?

Ruby on Rails is an open source web application development framework written in the Ruby programming language. It makes developing web applications faster and easier by making assumptions about the different steps required by any web programmer. By doing so it is able to automatically create necessary file structures and basic code, substantially decreasing the start-up time for an application. Furthermore, the framework created by Rails makes customizing an application easy after it's initial setup.

Ruby on Rails was developed around three basic principles. The first of these is DRY, or Don't Repeat Yourself. This is the idea that writing the same code over and over again should be avoided. Rails thus keeps code clean and in sync by making sure it's all kept in a single, unambiguous place. The second, and very closely related, principle is Convention Over Configuration. Rails makes assumptions about how an application will be developed, reducing the amount of code and number of configuration files needed to make all the components work together. For example, naming conventions are used to map database tables to their models without requiring an explicit specification of this connection. Then, in conjunction with the DRY principle, table columns can be accessed using the model class without actually being defined within it. The last principle of Rails is to use the REST architecture, a.k.a. Representational State Transfer, in developing web applications. The REST architecture consists of a set of principles that define how web resources should be used. These include standard addresses and interfaces for accessing resources, as well as a stateless client-server data transfer protocol. Together these principles create a development framework that vastly simplifies the work needed to be done to build a web application.

A History of Ruby on Rails

Ruby on Rails is based on the language Ruby which came out of Japan in 1995. The Ruby on Rails framework was first created by David Hansson and was released as open source in July of 2004; however, the commit rights for the Rails framework were not opened until February of 2005. On December 23rd, 2008 another model-view-controller web framework tool called Mongrel+Erb or Merb was launched. This was merged with Ruby on Rails causing drastic differences between the 3rd version of Rails and its previous versions, thus eliminating unnecessary duplication between the two frameworks.

Ruby on Rail Basics

Ruby on Rails runs on all major operating systems, including Windows, OS X (Rails ships with OS X), and Linux, and contains all the parts necessary to build a web application. Rails applications need to be run on a web server. While WEBrick and Mongrel are the servers most often used with Rails, most others such as Apache, Lighttpd, Abyss, and nginx, can also be used. A Rails application usually interacts with a database, thus Rails supports most common database management systems, including MySQL, PostgreSQL, SQLite, and Oracle. Applications can be developed using the console and a simple text editor, or an IDE such as RadRails (which is a plug-in for Eclipse).

Ruby on Rails uses the MVC (Model-View-Controller) architecture to organize an application. Using an MVC approach is beneficial because it separates the different program components and keeps code compartmentalized. The models are used to represents the application data and usually correspond to database tables. The views describe the user interface and consist mainly of HTML with embedded Ruby code. Finally, the controllers link the models and views by supporting their interactions.

There are six main packages in the Rails framework, each of which is responsible for different web application components. These are:

- Action Pack - Action Pack contains the Action Controller, Action Dispatch, and Action View packages. Action Controller and Action View manage the controllers and views, respectively. Action Dispatch is used to handle routing web requests.
- Action Mailer - Action Mailer is used for developing e-mail services.
- Active Model - Active Model provides an interface between the Action Pack components and the ORM (Object Relational Mapping) system Active Record (see below).
- Active Record - Active Record is the ORM system used by Rails. It is used to map database information to usable objects. See below for details about ORM and Active Record.
- Active Resource - Active Resource is used to manage the interactions between local application objects and RESTful web services.
- Active Support - Active Support contains all the necessary utility classes and Ruby library extensions used by Rails.

There are a number of different components used by Rails to facilitate easy web development. One of the major components is generators. Generators are scripts that automate common development tasks such as model creation. They create the appropriate files and base code for these actions. Generators are the main way to get an application up and running, and will be discussed in more detail later. Another way in which Rails speeds up web development is by performing scaffolding. Scaffolding is the automatic creation of operations and views for database tables. Scaffolding generates primary code for these actions, which can later be customized for the particular application. Finally, Rails allows databases to be altered using

migrations. Migrations are a mechanism for reversible and traceable changes to be easily made to the databases. All of these Rails components mean that the developer rarely has alter or access the database by writing SQL code. This makes database interaction easier and also reduces the number of errors that can occur in the transfer of information between the database and the web application.

Object Relational Mapping (ORM) and Active Record

ORM, or Object Relational Mapping, is a technique used to map database tables to their corresponding run-time objects. It abstracts the implementation details of database manipulation. Rails accomplishes ORM with Active Record, which essentially allows database tables to be represented as model classes. The standard (non Rails) way of accessing a database is to grab the result set using a SQL statement. By utilizing Active Record, database information can be obtained without using any SQL (though it is possible to do so for more complex queries).

If the Ruby on Rails naming conventions (Model is singular, database table is plural) are followed, then the association between the model object and the table will be established automatically by Active Record. Active Record also maps the table rows to objects, and columns to attributes.

Active Record implements table relationships by recognizing foreign keys, one-to-one associations, one-to-many associations, and many-to-many associations. For example, if we have to model employee, department, and managers, we can define the relationship simply using `employee belongs_to department`, `department has_one manager`, `manager has_one department`. We can access the manager of an employee just by using `employeeName.department.manager`, rather than having to query the database table for foreign keys directly. This implies that we can use the model directly rather than having to write explicit SQL queries. In the event that a custom SQL query is required, Active Record will escape all special characters in the SQL statement, which defends against SQL injection attacks, such as when a user's input is used within an SQL statement.

Active Record also performs a number of other functions for models in Rails. These include data validation, such as checking the presence or format of attribute values when creating or editing objects. It also allows for inheritance hierarchies and the creation of model classes consisting of object aggregations. Furthermore, Active Record mediates an object's transactions with the database, allows direct database manipulation, and provides methods for using callbacks.

Object Relational Mapping should not be confused with object-oriented databases. ORM is a technology that maps an object to a relational database. In the rare case that the Active Record is not sufficient, it is possible to write custom SQL queries.

Generator Functions

Ruby on Rails enables a developer to quickly get a project up and running by using convention over configuration. The variety of generator functions is perhaps one of the greatest tools to get a working web application quickly. The following is a brief overview of four of the most helpful generators that the Rails framework has to offer.
The model generator will create the migration file which will then create the database table for the new model when rake db:migrate is executed. The newly created model is an Active Record object which also enables easy interaction with the underlying database without explicitly using an SQL query. A unit test file is also created to write model tests in.

Model Generator (syntax: `rails generate model forum title:string content:text`)
The model generator will create the migration file which will then create the database table for the new model when *rake db:migrate* is executed. The newly created model is an Active Record object which also enables easy interaction with the underlying database without explicitly using an SQL query. A unit test file is also created to write model tests in.

Controller Generator (syntax: `rails generate controller forum`)
Ruby on Rails also has a generator function to create a controller. This will create the controller Ruby file along with an empty view directory for the *html.erb* files that will be rendered to the browser. A helper function file is created which allows the programmer to move any non-trivial logic that would otherwise clutter the view into this file, which leads to cleaner, more readable code. The controller generator also makes the assumption that tests will be written and will create test files for that purpose.

Scaffold Generator (syntax: `rails generate scaffold forum title:string content:text`)
The scaffold generator is ideal to use when you want to create the model along with the corresponding view and controller. This is like a combination of the model and the controller generators previously described; however, the scaffold generator will automatically create controller actions for a full set of CRUD (create, retrieve, update, delete) operations along with the corresponding views. Though this automatically generated code should likely be replaced by application-specific code, this will get a project up and running with minimal effort. This is a great example of the Rails philosophy of convention over configuration.

Migration Generator (syntax: `rails generate migration add_comment_to_forum`)
In every project there comes a time that you need to modify the database schema, which can become complicated and if mistakes are made it can take a lot of effort to revert to a previous version of the schema. The migration generator greatly reduces the complexity and time required to perform a database migration. This generator function will create the specified migration file which can then be edited to reflect whatever database change is required.
There are two parts to a migration file, `self.up` and `self.down`. The `self.up` function is the migration that you are performing and the `self.down` function will reverse the migration. This allows programmers to easily move to different versions of the database by using the various `rake db:migrate` commands. Put another way, using migrations allows for incremental changes to the database and also allows for rolling back to a previous version.

XML

XML is the preeminent standard for data exchange among web applications. Ruby on Rails not only supports XML, but makes it transparent and easy.
Exporting data in XML format from a Ruby on Rails web app is as easy as appending the ".xml" file extension to the URL for a given page. For example, http://localhost:3000/blog/posts/1 will present the user with an HTML view of the first post of the blog hosted at this address; http://localhost:3000/blog/posts/1.xml will give the XML representation of the underlying data, without the presentational markup.
At the heart of this ability is the fact that any Model object in Rails (and remember, Rails provides these Model objects for free) has a built-in `to_xml()` method. This data could then be passed into another application as input, or into an external parser for transformation or further processing.

Similarly, XML can be embedded as part of an HTTP response by creating an appropriate method within the Controller object for a given resource, that in turn calls the built-in `render()` method with arguments to make it return the XML representation of the data.
(Of course, one could also simply override the `to_xml()` method in the Model object, but separating concerns by putting this in the Controller object allows greater maintainability, flexibility, and extensibility.)

Rails can also process or transform that initial XML representation itself, with little programming effort. This would allow interaction with third-party applications which expect any XML input data to be in a specific format over which a user has no control - for example Google Maps.
Simple modifications to the XML are just a matter of adding a few options to the Controller's call to the `render()`/`to_xml()` methods.

Such simple manipulations include omitting certain elements (for example, some metadata about a record that your application maintains for internal use but that a third-party application does not care about), or specifying a new root element (for example, your application has blog posts rooted at the <post> node, but a third-party application expects a root of <data>).

Deeper transformations that change the structure of the XML, such as moving elements between levels or grouping elements within other elements, are also straightforward with the use of an XML Builder Template. The details are beyond the scope of this report, but the high-level summary is that basic programming constructs can be used to map the underlying data into a new XML structure, by extracting the pieces of the original XML and inserting them into appropriate locations in a new document.

These facilities in Rails make it easy to add to your web application features like a mashup with Google Maps, or an RSS feed. The power is there to do whatever you want - easily.

Security

Web applications are vulnerable to many different types of attacks such as SQL injection. Active Record, employed properly, will protect against SQL injection attacks. Cross-site scripting is another common attack that is performed against a websites. Ruby on Rails 3 is able to defend against such an attack by assuming that everything outputted to the browser is unsafe, so `<script>` tags are automatically escaped. Form forgery is another attack that Rails 3 provides protection against by utilizing authentication tokens in all of its forms.

Disadvantages of Ruby on Rails

Ruby on Rails falls short in some regards when compared with PHP; a general purpose scripting language that is widely used for web development. Ruby on Rails runs slower and does not scale as well as PHP. An example of the scalability problem is shown with Twitter, a Ruby on Rails website, which has had several outages. Another problem with Ruby on Rails is that it is not easily deployed on a cluster. It also lacks support in a Windows environment and thus runs better on a Unix-type operating system. Additionally the ORM can conflict with legacy databases. Ruby on Rails works from the middle-ware outwards. It is Rails' generator functions that allow a developer to get their website up and running quickly; however, this can conflict with a database schema that was created before. Rails has its own naming conventions, for instance naming the table in a pluralized form, which can conflict with previously user-defined schema.

Advantages of Ruby on Rails

There are many advantages to using Ruby on Rails. Generator functions are a major advantage as they enable a developer to get a web application up and running quickly by utilizing automatic code generation. Another aspect of Rails which leads to quicker development time is the Rails convention over configuration principle which means that the programmer does not have to take time specifying minute details in configuration files. This occurs because Rails makes an assumption about what the programmer is doing and how the task should be carried out. The design pattern, model-view-controller, that Rails employs is another advantage to Rails framework as it separates the logic from the user interface which makes it easier to avoid code duplication and makes code clearer. Overall this design pattern makes for easier maintenance and thus takes less time. Additionally, Rails has lots of support for HTML and it also automatically carries out text validation within the forms by changing the CSS to highlight the invalid entries. Rails makes exporting to XML simple. There are plug-ins available in Ruby on Rails to extend features and eliminate repetitive code allowing the developper to focus on other parts of their web application. Overall Ruby on Rails is a great framework that enables a developer to get a working prototype or web application up and running quickly.

Conclusion

In summary, Ruby on Rails 3 is an impressively complete framework for developing web applications quickly and easily. Though it is relatively young it is still mature, stable, and widely used. Rails has a rich tool-set which makes the most common use cases in web-application development as simple as a single command. Ruby on Rails shines especially in rapid

prototyping, letting you create a functioning web application with minimal effort. But it also has the power and flexibility to evolve your projects into full maturity.

Websites that use Ruby on Rails

- Twitter (http://twitter.com/)
- Groupon (http://www.groupon.com/)
- Shopify (http://shopify.com/)
- Github (http://github.com/)
- Yellow Pages (http://www.yellowpages.com)
- Hulu (http://www.hulu.com)
- Scribd (http://www.scribd.com/)
- Chow (http://www.chow.com/)
- Urban Dictionary (http://urbandictionary.com/)

Resources

1. Ruby A Programmer's Best Friend. <http://www.ruby-lang.org/en/>.
2. Ruby on Rails. <http://rubyonrails.org/>.
3. Ruby on Rails. <http://en.wikipedia.org/wiki/Ruby_on_Rails>.
4. The Top 20 Plugins to Create a Rails Application. <http://blog.devinterface.com/2010/04/the-top-20-plugins-to-create-a-rails-application/>.