Web Application Architectures

Module 2: Ruby on Rails
Lecture 4: Rails Philosophy

THE UNIVERSITY of NEW MEXICO

Rails Philosophy

The Rails philosophy is based upon three principles:

Convention over Configuration – Common aspects of a web application are provided (i.e., preconfigured) for you, so use them, rather than fight against them! Ideally, the developer should only have to specify the unconventional aspects of the application.

Don't Repeat Yourself (DRY) – Every piece of information should have a single, unambiguous, authoritative representation within a system. Duplication of code fragments throughout an application can lead to logical contradictions, and in general make the application more difficult to maintain.

Agile Development – Software development methodologies based on iterative and incremental development, where requirements and code evolve with minimal planning through self-organizing, cross-functional teams.

Convention over Configuration

THE UNIVERSITY of NEW MEXICO

- A massive number of "conventions" are built into Rails, and the real trick is learning all of them. When you create a Rails application, the full web stack is "pre-wired" and ready to go.

- Rails code generators follow specific naming conventions. E.g., if you use the scaffold (MVC) generator to create a `Post`, you get:
  - A class called `Post` will be created for the model, along with a corresponding table in the database that will be called `posts`
  - A RESTful controller called `posts_controller` will be created, and routes will be set up so that specific URLs will be able to perform CRUD operations on the `post` table in the database.
  - A `posts` view will be created, consisting of a set of HTML files that can be used to render the results of the CRUD operations.
  - Test fixtures, along with unit, functional, integration and performance test suites are automatically generated.

DRY

Every piece of information should have a single, unambiguous, authoritative representation within a system.

E.g., In Rails, with ActiveRecords, once a model is specified, you don't need to specify database column names –they're determined from the model.

- When applied successfully, a modification to a system element does not change any other logically-unrelated system element, while elements that are logically related all change predictably and uniformly, thereby keeping them in "sync".
- This principle makes it easier to use code generators and automatic build systems.
- Thus, DRY code is typically created by data transformations and code generators, allowing the software developer to avoid copy and paste operations.
- Following the DRY principle makes it easier to maintain large software systems.

Agile Development

- Agile development emphasizes working software as the primary measure of progress.
- Rails was built with agile development in mind:
  - A working application is available immediately.
  - In development mode, there are no recompile, deploy, restart cycles. I.e., Rails does not generally require you to stop the server; changes made to the application will be automatically picked up by the server.
  - Rails has simple tools to generate code quickly.
  - Testing is built into the Rails framework.
- Extreme programming is an agile approach that that centers around test-driven development (TDD). Behavior-driven development (BDD), a second generation agile approach, extends TDD by writing test cases in natural language that non-programmers can read. BDD focuses on obtaining a clear understanding of desired software behavior through discuss with stakeholders.
- RSpec and Cucumber are BDD tools for Ruby that we'll use.