УДК 004.415.2:004.652

ПРИМЕНЕНИЕ ПАТТЕРНОВ ПРОЕКТИРОВАНИЯ В RUBY ON RAILS 3 НА ПРИМЕРЕ ПРОСТОГО ВЕБ-СЕРВИСА

Дощанов Ж.А.
Евразийский национальный университет им. Л.Н.Гумилева, Астана

Научный руководитель – Баймуратова Г.Г.

Паттерн (шаблон) проектирования - описание взаимодействия объектов и классов, адаптированных для решения общей задачи проектирования в конкретном контексте. Паттерны проектирования упрощают повторное использование удачных проектных и архитектурных решений [1].

Веб-сервис (веб-служба) — идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах (XML, JSON и т. д.)

В данной статье будет рассмотрено использование паттернов проектирования в современных веб-фреймворках на примере Ruby on Rails 3.

Ruby on Rails – фреймворк SaaS приложения, который определяет конкретную структуру для организации кода приложения и предоставляет интерфейс для сервера приложения. Сервер ждет запрос веб-браузера и ставит в соответствие каждому входящему запросу (URI и HTTP метод) конкретный метод в одном из контроллеров приложения [2].

Начнем обзор простого веб-сервиса на Ruby on Rails. Все данные хранятся обычно в базе данных, причем надо отметить, что в приложении нет сильной привязки к СУБД. Когда клиент совершает некоторый запрос, его обрабатывает определенный для этого запроса метод в контроллере. Метод и контроллер выбираются на основе URI и HTTP метода запроса. Данный подход носит название REST (Representational State Transfer). Он гласит, что интерфейсы программирования приложения (API) должны давать доступ к независимым операциям [2]. В контроллере запрос проходит обработку, во время которой контроллер может обратиться к модели – объекту приложения, предоставляющего доступ к БД. Завершающей стадией является возвращение ответа контроллера клиенту. Ответ может быть представлен как в виде HTML страницы, так и в виде XML или JSON. В случае возвращения HTML страницы ее вид мы можем определить с помощью представления (View).

Архитектура, в основе которой лежат модель, представление и контроллер, получила название MVC (Model-View-Controller). MVC отделяет вид от модели, устанавливая между ними протокол взаимодействия «подписка/оповещение». При каждом изменении внутренних данных модель оповещает все зависящие от нее представления, в результате чего представление обновляет себя. Такой подход позволяет присоединить к одной модели несколько различных представлений [1]. Функции представления заключаются лишь в графическом представлении данных. Все изменения данных обрабатываются контроллером. Контроллер получает входные данные, выполняет операции над моделью и указывает представлению на необходимость обновления [3]. Большая часть логики приложения регулируется контроллером.

Роль контроллера в Rails выполняет класс, являющийся наследником класса ApplicationController. Его основными функциями являются поддержка сессии, фильтрация и кеширование. URI и HTTP метод ставятся в соответствие методу контроллера в файле /config/routes.rb. Метод контроллера может как вернуть представление, так и перенаправить на другой метод с некоторыми параметрами. Суперклассом каждого

представления является класс ActionView. Представления хранятся в папке views, каждому представлению относится один файл. Формат файла и синтаксис могут различаться в зависимости от используемого языка разметки шаблона (erb или haml). Наиболее часто представление используются в качестве шаблонов (templates), макетов (layouts) или помощников форматирования (helpers). Модели расположены в папке models, каждая модель привязывается к определенной таблице БД.

Коснемся того, как в Ruby on Rails происходит обращение к данным. Модель и данные связываются с помощью паттерна Active Record (активная запись). Active Record – объект, выполняющий роль оболочки для строки таблицы или представления базы данных. Он инкапсулирует доступ к базе данных и добавляет к данным логику домена. Этот объект охватывает и данные и поведение. Большая часть его данных является постоянной и должна храниться в базе данных. В типовом решении Active Record используется наиболее очевидный подход, при котором логика доступа к данным включается в объект домена. В этом случае все знают, как считывать данные из базы данных и как их записывать в нее [3]. Реализация Active Record в Rails основана на том же подходе. Выше упоминалось, что в Rails нет сильной привязки к СУБД. Вы можете подключиться к базе данных, используя **API**, предоставляемый Active Record, который создает необходимый запрос непосредственно в движок БД при помощи адаптеров. У Active Record есть адаптеры для MySQL, Postgres, MS SQLServer, DB2, и SQLite. Необходимо лишь записать параметры доступа к БД в файле database.yml. С помощью статических методов класса модели можно выполнять базовые операции поиска и CRUD (Create Read Update Delete – Создание Чтение Обновление Удаление), кроме этого можно определить собственные методы, которые будут реализовывать некоторые фрагменты бизнес-логики приложения. Надо отметить интересную возможность Rails – ассоциации в Active Record, позволяющие декларативно выражать отношения между классами моделей. Велика роль валидаторов в Active Record. Они позволяют в декларативной форме объявлять допустимые состояния объектов модели. Процесс разработки упрощается также возможностями самого языка Ruby (например, method chaining).

Подводя итог, можно выделить основу архитектуры Ruby on Rails:

- Active Record.
- Action View.
- Action Controller.

Сочетание последних двух известно как Action Pack.

В данной статье были рассмотрены паттерны MVC и Action Record, а также их реализация на фреймворке Ruby On Rails 3. К сожалению, мною не были освещены шаблоны проектирования TDD (Test Driven Development) и BDD (Behavior Driven Development), используемые в Rails т.к. каждая из этих тем крайне широка и заслуживает отдельной статьи. В целом, применение вышеперечисленных паттернов делает фреймворк Ruby on Rails удобным и гибким инструментом для создания масштабируемых и легко тестируемых веб-приложений, в особенности веб-сервисов. В реалиях современного мира роль паттернов, часто используемых в веб-приложениях, становится все выше в связи с постепенным переходом приложений в Интернет. Хотелось бы отметить, что шаблоны MVC и Active Record также лежат в основе нового популярного фреймворка ASP.NET MVC, выпущенного компанией Microsoft в 2007 г. с целью составить конкуренцию Ruby on Rails. Отсюда можно сделать вывод, что многие современные архитектуры строятся на одних и тех паттернах проектирования.

Литература

1. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб: Питер, 2010, 368 с.

2. Fox A., Patterson D.. Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing. – Strawberry Canyon LLC, eBook, 2012, 310 c.