+32

lolmaus *(комментарий был изменён)*#

↑

Развиваю на работе проект на Angular, а также создал и развиваю проект на Ember. Также есть опыт Backbone.

Разница между Angular и Ember — просто небо и земля, в пользу Ember.

***

У Angular очень низкий порог вхождения, но очень высокая сложность использования. Сложность в том смысле, что крайне сложно сделать по-настоящему качественно. Выбрав Angular, вы наступите на все возможные грабли, подложите себе множество мин замедленного действия. Рефакторинг для вас будет как ремонт в советской квартире — процесс постоянный и мучительный. Нужно пройти огромный (полагаю, многолетний) путь, чтобы достичь такого уровня владения Angular, который позволит написать приложение с действительно качественной кодовой базой с первого раза.

Angular очень низкоуровневый. В нём всё делается на коленке. Радоваться такому подходу могут только те, кто никогда не пробовал добротный высокоуровневый фрэймворк и кому не приходилось поддерживать проекты, писаные несколькими сменами говнокодеров. На самом деле это полная жопа. Чтобы писать качественный, читаемый и поддерживаемый код на Angular, надо иметь строжайшие гайдлайны и соблюдать их. На деле же всё пишется как быстрее, ведь Angular так легко позволяет срезать углы.

Сообщество Angular — это как сообщество PHP. И то, и другое — это губки, которые впитывают всех дилетантов и бесталанных разработчиков. Подчеркиваю, что я не утверждаю, что толковых и талантливых разработчиков там нет. Есть, конечно, но их процентное соотношение — самое низкое среди всех фрэймворков/языков.

Angular — это не MVC-фрэймворк. Они долгое время позиционировали себя как «MV*», потом они убрали эту аббревиатуру с сайта. Ржака в том, что в Angular нет ни View, ни Model, по сути, Angular — это голый контроллер. Вместо слоя модели можно использовать что-то свое, например, JSData, полностью слизанный с Ember Data. Но в 99% поделок на Angular модель как отдельная сущность отсутствует, данные хранятся, как бы это сказать, в ладошке. Со слоем представления всё печально, так как в Angular его нет, но и взять свой нельзя. Вместо вьюх Angular использует DOM нагорячую. То есть Angular берет существующий DOM, проходится по нему и модифицирует на лету. Вам приходится вставлять Angular-разметку прямо в HTML, и вся эта разметка попадает в браузер. HTML-код Angular-приложения в инспекторе браузера выглядит закиданным яйцами и туалетной бумагой, и у подавляющего большинства Angular-приложений HTML-код невалиден.

Логика в HTML! Это просто цирк какой-то. В принципе, Angular не обязывает вас выносить логику в HTML, но вы не найдете ни одного приложения на Angular, которое этим не грешит.

Angular трудно тестировать. Angular заявляет, что написан для удобного тестирования, но на практике вы столкнетесь с трудностями и значительную часть кода оставите непокрытой тестами.

Документация Angular — это что-то с чем-то. В разделе Services приводятся примеры использования `.factory` и нет `.service`, а примеры использования `.service` приводятся на странице Providers. Оглавлений у простыней страниц нет, найти что-то сложно. Если не верите на слово, попробуйте в официальной документации найти, какие аргументы принимает колбэк `link` у директивы (подсказка: все страницы с упоминанием `link` или `directive` в заголовках можете пропускать, там не найдете).

Я уже не говорю про проблемы с производительностью Angular. Это просто курам насмех. 2000 binding'ов — и canvas-анимации начинают подтормаживать. Становится видно, как Angular скрипит шестеренками даже когда пользователь ничего не делает. Чтобы снизить количество binding'ов, приходится прибегать к методам работы, применяемым в Backbone. Это крайне досадный шаг назад.

***

Ember — полная противоположность Angular. Освоить его сложно. Мне потребовалось два месяца, чтобы почувствовать себя в нем комфортно. Но с этого момента пользоваться им стало очень и очень просто, удивительно удобно и продуктивно.

Прежде всего скажу, что Ember очень гибок. Его «предвзятость» ни в чем вас не ограничивает. Не верьте тем, кто говорит, что Ember вас ограничивает в написании вашего приложения (так пишут только те, кто перепробовал фрэймворки для галочки, не задержавшись в каждом дольше чем на день). Всё, буквально всё в Ember расширяется и переопределяется. Более того, фрэймворк изначально построен с расчетом на то, что вы будете его расширять и переопределять.

«Жесткость» Ember проявляется только в структуре вашего проекта. Стандартизованная структура, благодаря подходу «договоренностей вместо конфигурирования» (convention over configuration), избавляет вас от необходимости писать обслуживающий код. Изначально, Ember использует дефолтные сущности для всего, что ему требуется (вьюхи там, контроллеры, шаблоны, маршруты...). Стоит вам сохранить кастомную сущность под соответствующим именем, как Ember начинает использовать ее вместо дефолтной. Таким образом, вы пишете только бизнес-логику: описываете суть вещей, без «воды». Из всего, что я писал на JS, код на Ember получается самым компактным и наиболее экспрессивным. По мере развития вашего проекта его структура не превращается в помойку. Структура, кстати, не такая уж и жесткая: можно группировать файлы по типу сущности (models/user.js, components/user.js), а можно по названию сущности (user/model.js, user/controller.js), выбирать вам.

Ember — это полноценный MVC-фрэймворк. В числе прочего, у него самый лучший в мире роутер и на редкость грамонтный слой модели.

В Ember используется свой сборщик (на основе Broccoli, это конкурент Gulp) и готовая структура проекта. Они очень качественные, очень богаты функционально и очень активно развиваются. Но самое главное, сборщик и структура проекта — стандартны. Все проекты построены на одних и тех же конвенциях, никто не изобретает убогие велосипеды с костылями.

Ember побуждает грамотно структурировать код. В Ember нужно постараться, чтобы напортачить — в отличие от Angular, где приходится постоянно тужиться, чтобы НЕ напортачить. После короткого вводного курса по Ember новобранец способен писать

приличный, читаемый и поддерживаемый код.

В Ember из коробки идет поддержка ES6, можно сразу писать современный код. BabelJS обеспечивает обратную совместимость (до IE8 включительно). Используются чистые, лаконичные ES6-модули, которые, пока браузеры не обзаведутся их поддержкой, транспилируются в AMD.

В Ember очень удобная система аддонов. Аддоны можно писать как для самого Ember, так и для сборщика. Аддоны распространяются через npm, который по ряду критериев практичнее Bower'a. Например, npm позволяет лочить версии зависимостей при помощи lock-файла (shrinkwrap), хотя до практичности Ruby Bundler им обоим как до луны.

В Ember очень удобная система классов. Оставаясь интуитивно понятной для JS-кодеров, она обладает многими преимуществами более благородных языков. Классы можно расширять динамически (причем, как инстансы, так и классы), из переопределяющего метода можно вызывать переопределяемый. Есть mixin'ы. В общем-то, значительная часть функционала вынесена в mixin'ы, плюс идет процесс рефакторинга: всё больше базового функционала Ember выносится из встроенных классов в mixin'ы, облегчая переиспользование кода. Например, для типового формирования URL'a раньше надо было require'ить сериализатор, что увеличивает связность приложения, а это не очень хорошо (tight coupling). А теперь можно в любую сущность подключить mixin с методом формироавния URL'a.

Шаблонизатор Handlebars — это просто благословение, особенно после Angular'овского засирания DOM. Возможности шаблонизатора намеренно ограничены (как в Django), чтобы не допустить вынос логики во вьюхи. Движок шаблонизации в Ember активно развивается, недавно он вышел на качественно новый уровень, позаимствовав некоторые приемы из React. Вот тут React прикрутили в качестве view layer в Ember, и можно сравнить скорость рендеринга сложной вьюхи между React и чистым Ember: github.com/ghempton/ember-react На глаз, разница примерно двукратная: где-то 0.2 и 0.4 секунды. А со следующим развития шагом Ember догонит и, возможно даже, перегонит React, см. github.com/emberjs/ember.js/pull/10501. К слову, в инспекторе браузера HTML-код Ember-приложения выглядит почти полностью чистым (добавляются айдишники), и он валиден. А для любителей indendation based синтаксиса есть целых два препроцессора: один с синтаксисом по примеру Jade/Slim, другой по примеру Haml.

Вычисляемые свойства Ember позволяют писать реактивный код. За пару месяцев использования Ember в вашей голове происходит смена парадигмы с императивной на декларативную. Это очень круто и удобно. При том же объеме функционала, код становится короче и яснее. Кроме того, вычисляемые свойства ленивы: они не вычисляются до тех пор, пока не будут запрошены, а результат вычисления запоминается. Повторное вычисление происходит только при изменении свойств, от значений которых зависит значение данного вычисляемого свойства.

И никакого dirty checking, Ember использует observer'ы. Благодаря этому data binding не сказывается на производительности приложения и работают интуитивно понятно. Когда вы меняете значение свойства, Ember синхронно уведомляет все заинтересованные сущности об изменении. «Под капотом» цепочки вычислений группируются в так называемые run loops, в результате обновление DOM происходит пакетно в конце цепочки, а не на каждом шаге.

В Ember очень активное и дружелюбное сообщество. Не находясь в когтях корпорации,

Ember развивается в открытую. Все планы обсуждаются сначала в RFC'ах, потом в pull-request'ах. Главные мэйнтейнеры раз в N недель слетаются на встречу, чтобы обсудить проблемы развития в реале. Все мэйнтейнеры Ember — сотрудники различных «продуктовых» компаний, использующих Ember в продакшене.

Ember развивается по принципу «стабильность без стагнации». «Стабильность» здесь означает отсутствие такой жопы, как в Angular, где новая мажорная версия фрэймворка оказывается заведомо несовместимой с предыдущей. «Без стагнации» означает отсутствие такой жопы, как в Node, где развитие проекта заморожено в угоду поддержке корпоративных клиентов и вопреки нуждам сообщества. На практике этот подход означает, что, во-первых, новые фичи вводятся по мере готовности, не откладывая до мажорного релиза. Во-вторых, breaking changes вводятся с длительным deprecation period, давая пользователям возможность без нервотрепки поддерживать кодовую базу в актуальном состоянии. Релиз мажорной версии будет означать по сути просто отключение поддержки ранее deprecate'нутых фич.

В последнее время Ember рекомендует подход, аналогичный React: data up, actions down с использованием односторонних data binding'ов. Это делает структуру приложения более плоской и упрощает отладку. Однако вы можете использовать двусторонние data binding'и там, где считаете уместным (например, для input'ов). Я свое первое приложение построил целиком на двусторонних, и структура приложения получилась очень простой (всё обновляется и пробрасывается само, не надо ничего делать руками), при этом я не столкнулся с проблемами ни в производительности, ни в отладке.

В Ember самый функциональный и удобный инструмент отладки в инспекторе браузера. Angular Batarang стыдливо прячется за спинами.

Ember очень легко тестировать. Полная инфраструктура тестирования идет из коробки: всё преднастроено и готово к использованию. Сборщик создает заготовки тестов для каждой сущности, которую вы используете. Покрыть юнит-тестами можно абсолютно любую сущность, будь то модель, маршрут или даже какой-нибудь initializer. 100%-ное покрытие риальне! Очень легко писать приемочные (acceptance aka integration) тесты благодаря удобным хелперам, позволяющим работать с асинхронными операциями в синхронном стиле. В комплекте идет QUnit, но есть аддон, добавляющий поддержку Mocha, хотя благодаря хелперам особой нужды в Mocha нет. Также есть аддон для Chai. Статус тестов можно смотреть не только в консоли, но и в браузере во время работы dev-сервера. В процессе разработки приложение на Ember компилируется инкрементально, благодаря чему ждать перезапуска тестов приходится не дольше пары секунд. В комплекте идет конфиг для Travis: публичный проект на Github можно бесплатно тестировать на CI-сервере.

Документация у Ember достаточно хорошая. Есть замечательный официальный Guide, способный заменить книгу. Доки по API пишутся прямо в исходниках соответствующих компонентов при помощи специальным образом сформированных комментариев. Благодаря этому документация для каждой версии Ember формируется автоматически и никогда не устаревает.

На builtwithember.io можете посмотреть примеры свободных и проприетарных приложений, выполненных на Ember. Там можно найти всё: разнообразные админкии, дэшборды, чаты, инструменты визуализации и анализа данных, CMS, CRM, соцсети, форумы, таск-менеджеры, трейдинговые площадки, средства рассылок, continuous integration, базы знаний, поисковики, платежные системы… Короче говоря, всё, что можно придумать. Из крупных

проектов: Vine (100 млн уников в месяц), ZenDesk (их бэкенд обрабатывает 100 тысяч запросов в минуту), TravisCI, DockYard, Twitch (100 тысяч просмотров в минуту в пиковое время; был куплен Amazon'ом за миллиард долларов).

***

Backbone — это несерьезно. Это просто удобная обертка над jQuery и Underscore. Data-binding'ов нет, MVC нет и много чего нет. Чтобы сделать single-page applicaiton на Backbone, вам придется писать бескрайние простыни обслуживающего кода. Это очень утомительно (для программиста) и дорого (для работодателя). Сложность поддержки Backbone-приложения растет экспоненциально по отношению к размеру кодовой базы.

PS С удовольствием отвечу на вопросы по Ember. Общие вопросы задавайте здесь в комментах, конкретные вопросы по коду пишите на StackOverflow (с примером на JSBin.com) и кидайте ссылку в личку.

+1

aen#

↑

Всяк кулик свое болото хвалит.

+7

lolmaus#

↑

Гм, вы так говорите, как будто у меня не было выбора. Как будто я был вынужден использовать только Ember и хвалю его из зависти к Angular и React.

Напротив, я тщательно сравнил акутальные на тот момент фрэймворки: Angular, Backbone, Knockout, Ember, Derby (и ряд менее известных: CanJS, BatmanJS и т. п.) и сделал взвешенный выбор.

Derby показался наиболее перспективным. Но вместе с рядом ярких преимуществ: изоморфность, синхронизация и разрешение конфликтов из коробки, шаблонизатор принимает более одного блока при вызове компонента — есть и серьезные недостатки: медленное развитие, полное отсутствие какой-либо документации и справочных материалов за исключением Readme на титульной странице, маленькое сообщество, отстутствие тестирования из коробки, незрелость, выливающая в постоянные breaking changes.

Angular невзлюбил заочно. Когда я ознакомился с приемами ее работы и прошел туториалы, мне стало противно пользоваться им дальше. Уже одних только засранного DOM и отсутствия внятного шаблонизатора мне было достаточно, чтобы вычеркнуть его из списка. Однако когда на работе встал вопрос, на каком фрэймворке переписывать проект, я выбрал Angular. Мне важно было принять решение в пользу компании, а не себя любимого. Angular победил с большим отрывом по распространенности, документированности, количестве доступных специалистов.

Я надеялся, что, познакомившись с Angular ближе и на примере опытных спецов (проект пилили аутсорсерыcsssr.ru), я его если не полюблю, то хотя бы приму. Увижу, как принято на нем работать, освою приемы, и перестану испытывать к нему брезгливость. Как бы не так. После глубокого знакомства с Angular я в нем еще больше разочаровался. Сейчас развитие проекта целиком легло на мои плечи, и я проклинаю тот день, когда сел за баранку этого пылесоса.

Админку для нашего бэкенда (с достаточно сложным аналитическим функционалом) я написал на Ember и был поражен разительным отличием. Я давно интересовался Ember, но по одним только туториалам освоить его было сложно, и наконец, появилась возможность применить его на боевом проекте. Освоение Ember — это лучшее, что произошло в моей жизни после рождения сына. :D Мир снова заиграл красками, появился энтузиазм. :)

Так что не надо тут про кулика и болото. Сами-то в каком болоте сидите?

PS Посмотрел предыдущий коммент оратора, понял, что это тролль. Зря распинался, ну ладно.