сегодня в 01:19

Счет на оплату. Рабочее приложение на sails.js, ractive.js, Backbone.js

tutorial

JavaScript*, Веб-разработка*

Доброго дня, на выходных от скуки и отсутствия работы решил себя развлечь написанием небольшого приложения, которое сгодится в качестве учебного метариала для изучения возможностей двух замечательных библиотек — ractive.js и sails.js

Постановка задачи

По работе часто приходится после выполенения очередного задания (я — фрилансер) выставлять заказчику счет на оплату услуг. Тем более если имеешь дело с юридическими лицами. Для этого я использовал простой html-шаблон, в который данные заносил руками, исправляя очередные <td></tr>…

Выглядит примерно так

Признаюсь, стили и разметка угнаны с freshbooks.com, который я использовал в свое время. К сожалению, для русских клиентов он мне не подошел, да и простого html-шаблона мне хватало.

Выбор технологий

В текущем тренде популярности js-фреймворков всех мастей и серверной js разработки я хотел для этой задачи использовать нечто вкусное и реактивное, дабы немного побыть в этом потоке js счастья… И паралельно опробовать эти игрушки.

После недолгих изучений, сравнений и интуитивных озарений остановился на sails.js в качестве сервера. Выбирал между derby и sails — в итоге выбрал парусник, в основном из-за его простоты (дока читается легко и приятно), также в нем есть очень классный генератор rest api из коробки. Derby в плане изучения показался труднее и монструознее (для этого примера — явный оверхэд).

Лучшее за 24 часа

PrintBox3d. 3d-принтер по-русски

Автономные автомобили Google совершенствуют навыки вождения по городу

Вредные советы от создателей API Яндекс.Карт. Как сделать так, чтобы всё было плохо

Новый выделенный сервер: приемка и проверка

Работа с регистрами внешних устройств в языке C, часть 1

JSR 133 (Java Memory Model) FAQ (перевод)

Клавиатуры: раздельные, с ровными колонками и разворотом половин

Тренды в онлайн образовании

Новые разработки в области бионики позволяют бегать и танцевать

Счет на оплату. Рабочее приложение на sails.js, ractive.js, Backbone.js

все лучшие

Похожие посты

Обновленный облачный сервис Azure Web Sites для размещения сайтов PHP, Java, .NET, Node.js и Python 27.04.2014

NodeSchool, Node.js и один урок для самых маленьких 23.04.2014

Marionette.js. Drag&Drop сортировка моделей в коллекции 22.04.2014

Введение в Marionette.js Behaviors 21.04.2014

Matreshka.js v0.1 14.04.2014

Sails.js: первые шаги 11.04.2014

Node.js vs Ruby on Rails 11.04.2014

Игровой сервер за один день на Node.js + Socket.io 07.04.2014

Интеграция Passport в Sails.js 09.02.2014

jBone. Замена jQuery для Backbone или 2kb для DOM манипуляций 13.11.2013

На клиенте решил поиграться с ractive.js. И уже позже решено было подключить backbone.js — в основном из-за удобной работы с моделями.

До этого примера опыта **sails.js** и **ractive.js** у меня не было. В работе использовал только бэкбон.
Приступим.,

Сервер

Для нашего примера будем использовать sails v0.10 — она еще в стадии бета, но по сравнению с текущей стабильной версией 0.9.x в ней есть несколько плюшек, которые пригодятся. В частности model assocoations, которые позволяют задавать one-to-many, many-to-many (и другие связи между моделями), также в 0.10 переработана система grunt тасков. В доке по 0.10 все довольно ясно написано

sails v0.10 можно поставить через npm (я ставил глобально)
```
sudo npm install -g  "git://github.com/balderdashy/sails.git#v0.10"
```

проверяем
```
sails -v
```

0.10.0 — отлично

Создание скелета приложения sailsjs

Создаем новое приложение, например, invoicer и ставим зависимости
```
sails new invoicer
cd invoicer
npm install
```

Далее выполнив команду sails lift можно запустить встроенный express.js сервер на http://localhost:1337

Создание API сущностей (моделей)

Нам потребутеся 3 модели для приложения:

- user — для хранения данные о пользователе
- invoice — для списка счетов
- task — для задач в счете (инвойсе)

Создаем с помощью команды sails generate api <api_name>
▸ Генерация API

после этого в папке api/controllers появятся 3 файла
```
-rw-r--r-- 1  146 Апр 28 17:15 InvoiceController.js
-rw-r--r-- 1  143 Апр 28 17:15 TaskController.js
-rw-r--r-- 1  143 Апр 28 17:15 UserController.js
```

также api/models
```
-rw-r--r-- 1  146 Апр 28 17:15 Invoice.js
-rw-r--r-- 1  143 Апр 28 17:15 Task.js
-rw-r--r-- 1  143 Апр 28 17:15 User.js
```

Легко и просто sails создал для нас 3 метода,
http://localhost:1337/user
http://localhost:1337/invoice
http://localhost:1337/task
которые поддерживают CRUD операции. Также есть алиасы для них, например, http://localhost:1337/user/create?name=Andrey&address=Russia — создаст новый инстанс юзера. Можно поиграться через postman

Также советую ознакомится с документацией по контроллерам

Конфигурация хранилища (БД)

Где же хранятся созданные данные? По дефолту в качестве хранилища используется диск, что указано в настройках config/connections.js и config/models.js
▸ код config/connections.js

Мы же будет использовать mongo для хранения записей, для этого немного изменим config/models.js:
▸ код config/models.js

Опишем нужные нам поля модели User, Invoice и Task
▸ api/models/User.js

▸ api/models/Invoice.js

▸ api/models/Task.js

для использования монго адаптера нужно поставить пакет sails-mongo
```
npm install sails-mongo@0.10
```

Добавление `action` для контроллера, и шаблона (view) для него

Нам необходимо создать контроллер, который будет генерировать страничку для нашей основной задачи (создание инвойса):
```
sails generate controller main generate
```

Вопросы по теме

Как отправить POST из формы, используя Backbone?

Как сделать render таблицы в Underscore/Backbone?

Проблема с подключением Require.js (Не тянет темплейт)

Переменные Underscore-темплейт не принимают значения из метода fetch().(Uncaught ReferenceError: variable is not defined)

Как показать данные с backend (Flask) на frontend view (Backbone)?

Как использовать шаблонизатор Backbone.js в связке с Flask-backend?

Как реализовать аутентификацию backbone js + symfony2?

Backbone, функция extend и ее работа

Как организовать группировку коллекций и моделей?

Как реализовать подключение шаблонов на лету в backbone js?

Какая есть JavaScript библиотека для работы с API популярных сервисов (OAuth2, Social API, etc)?

Какую Javascript UI библиотеку/фреймворк использовать на замену Backbone/Marionette?

Как изучить backbone.js?

Как использовать разные шаблоны для роутов?

Как в JS работать с адресной строкой?

Как работает Backbone.Model.Fetch()(не обновляется модель)?

Какие есть полезные ресурсы и книги для начинающего frontend-разработчика?

Как решить проблему с фильтрацией коллекции backbone?

JS-шаблонизация на клиенте и доступность сайта для роботов: как лучше организовать?

Какой JS фреймворк выбрать для редактора фотоальбома?

Что обсуждают?

Безопасность базовых станций: выезд на измерения плотности электромагнитного поля 38

Настройка Apache для работы с СУБД Caché на Linux 3

Компания IBM представила новые серверы на основе процессоров Power8 5

Клавиатуры: раздельные, с ровными колонками и разворотом половин 11

Hello MongoDB (открытый удаленный доступ) 6

PrintBox3d. 3d-принтер по-русски 76

Вредные советы от создателей API Яндекс.Карт. Как сделать так, чтобы всё было плохо 53

Новые разработки в области бионики позволяют бегать и танцевать 5

Какие задачи решают IAMсистемы? 1

Автономные автомобили Google совершенствуют навыки вождения по городу 58

все посты

Компания дня

Стратоплан

Последний пост: 4 причины, почему

```
sails generate controller main generate
```

Мы создали новый MainController.js, в котором создана одна функция generate так называемый action
если перейти по урлу http://localhost:1337/main/generate мы увидим то, что нам вернула функция generate
По умолчанию она вернет json

```
return res.json({
    todo: 'Not implemented yet!'
});
```

Мы же хотим видеть в браузере html-страничку. Для этого вышеприведенный код заменим на

```
return res.view()
```

обновляем страничку в браузере и видим ошибку

```
{
  "view": {
    "name": "main/generate",
    "root": "/home/zaebee/projects/invoicer/views",
    "defaultEngine": "ejs",
    "ext": ".ejs"
  }
}
```

это значит что у нас не создан шбалон для view. Все htnl-шаблоны для контроллеров лежат в папке views и имеют следующую структуру
views/<controller_name>/<action_name>

создаем пустой шаблон views/main/generate

```
zaebee@zaeboo$ mkdir views/main
zaebee@zaeboo$ touch views/main/generate.ejs
```

По умолчанию в качестве шаблонного движка используется ejs. Sails поддерживает много шаблонизаторов и вы можете изменить его в файле config/views.js на ваш любимый:
ejs, jade, handlebars, mustache underscore, hogan, haml, haml-coffee, dust atpl, eco, ect, jazz, jqtpl, JUST, liquor, QEJS, swig, templayed, toffee, walrus, & whiskers

ВНИМАНИЕ! в версии sails 0.10 поддержка лайоутов работает только с ejs. Вкратце, есть базовый лейоут views/layout.ejs, от которого наследуются все остальные вьюхи. И при использовании шаблонизатора отличного от ejs наследования не будет. Sails дает это понять, если изменить опцию engine в файле config/views.js

```
warn: Sails' built-in layout support only works with the `ejs` view engine.
warn: You're using `hogan`.
warn: Ignoring `sails.config.views.layout`...
```

Клиент

Сервер готов, приступим к написанию клиентской части нашего приложения по создания инвойсов.

Подключение статики

Вся статика (или публичный клиентский код) лежит в папке assets. для того, чтобы поключить новые файлы к вашему шаблону просто поместите их в соотвествующую папку (скрипты в assets/js, стили в assets/styles, клиентские шаблоны в assets/templates) и sails с помощью своих grunt тасков запишет их в ваш index/layout.ejs — в специальные секции:
▸ Листинг исходного файла /views/layout.ejs

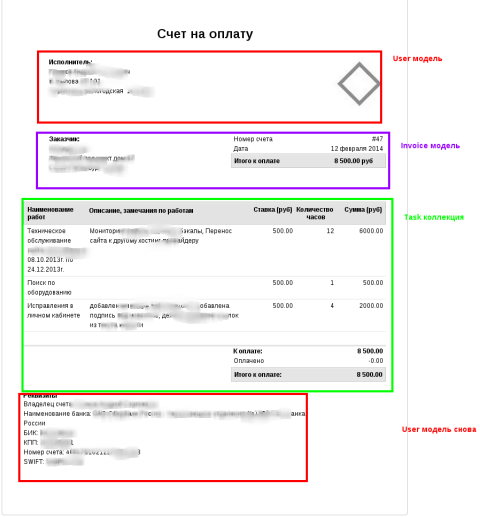
Подключим в наш layout нужные библиотеки (Jquery, Underscore, Backbone, Ractive) через cdn, такжк поместим bootstrap.min.css и готовый файл app.css в папку assets/styles. Также разместим дополнительные js либы, которые понадобятся (bootstrap.min.css, moment.ru.js и moment.min.js — библиотка для работы с датами) в папку assets/js/vendor и пустой файл app.js в папку assets/js. Запустим sails lift и посмотрим, что теперь у нас в файле views/layout.ejs
▸ Листинг исходного файла /views/layout.ejs

Отлично, sails сделал за нас, все что нужно. Правда, есть один минус — вендорские скрипты подключены ниже нашего app.js. Исправим файл tasks/pipeline.js укажем grunt`у, что папку vendor нужно подключать раньше:
▸ Часть листинга файла tasks/pipeline.js

Подготовка клиенской части завершена — можем приступать непосредственно к написанию бизнес-логики приложения.

Создание скелета разметки страницы. **Ractive.js** шаблоны

Взглянем еще раз на наш макет. На нем я выделил блоки, которые мы будем привязывать к нашим динамическим данным



Создадим базовую разметку в файле views/main/generate.ejs в которую будут инклюдиться наши клиентские шаблоны

▸ листинг файла views/main/generate.ejs

Итак, базовая разметка готова — пришло время для шаблонов **ractive.js**
Создадим для каждого нашего блока по шаблону (итого их будет четыре) и поместим их в assets/templates

▸ листинг файла assets/templates/invheader-upper.html

▸ листинг файла assets/templates/invheader-lower.html

▸ листинг файла assets/templates/invbody-tasks.html

▸ листинг файла assets/templates/invbody-account.html

В целом это обычный html, с вкраплениями mustache-подобных тэгов {{}}, в которых **ractive.js** вставляет свои данные. Также вы можете заметить некоторые директивы on-click="edit" — выполняет метод edit по клику; on-hover="toggleBtn", on-tap="destroy:{{this}}" этот момент осветим позже, можно пока изучить доку по евентам ractive.js

События подключаются в ractive в виде плагинов — так называемые proxy-events. Чтобы события заработали, нужно скачать нужные нам (я скачал все плагины для событий) и поместить их в папку assets/js/vendor
Поместим в эту же папку адаптер для Backbone, чтобы ractive.js смог использовать в качестве источника данных модели backbone.

Инициализация данных. Биндинг данных и шаблонов

Подведем промежуточный итог, что есть на данный момент и что мы хотим получить в итоге

- на сервере sails с помощью rest api позволяет создавать юзеров, инвойсы и задачи. Делать связи между ними за счет model associations. Данные хранятся в базе mongodb
- на клиенте backbone модели будут хранить введенные пользователем данные и сихнронизироваться с sails сервером через rest api
- на клиенте ractive будет осуществлять two-way биндинг между html-шаблонами и backbone моделями (за счет адаптера для Backbone)
- ....
- PROFIT?

для начала создадим нужные нам Backbone модели в нашем пустом файле assets/js/app.js:

▸ Листинг assets/js/app.js

Хорошо, теперь создадим ractive инстанс, который будет привязан к нашей модели app.User и будет рендерить наш шаблон assets/templates/invheader-upper.html и assets/templates/invbody-account.html
Создадим файл assets/js/user.js

▸ Листинг assets/js/user.js

Код достаточно прост. Здесь мы создаем базовый класс RactiveUser. Обычно можно создать инстанс через new Ractive({}), но в частности здесь нам нужно 2 элемента для пользователя, которые привязаны к одной модели и которые подписны практически на одинаковые события. Сами события указываются в теле init функции.

Едем дальше, создадим по аналогии assets/js/invoice.js и assets/js/task.js

▸ Листинг assets/js/invoice.js

▸ Листинг assets/js/task.js

Здесь также код достаточно понятен, для эвентов добавил комментарии. По сути это весь клиентский код. Планировал еще прикрутить метод для генерации статического инвойса на основе id (например, http://localhost:1337/main/generate/535ea7aa6113230d773fd160) или использовать api pdfcrowd.com, благо у них есть модуль для node, который позволяет по урлу создавать

pdf… Однако за выходные это сделать не успел. Сейчас я создаю pdf через ctrp+P (отправить на печать) -> «Печать в файл». А для того чтобы не вылезли ненужные html элементы (например, кнопки) — добавил для них класс hidden-print.

Деплой на сервер

На этом практически все — приложение готово. Данный пример находится на гитхабе

На сервере клонируем репозиторий, ставим зависимости и запускаем sails в продакшн режиме:

```
node app.js --port=8000 --prod
```

Запустил рабочее демо в продакшн режиме

Резюме

Итог работы как с sailsjs так и с ractive — очень порадовал.
Sailsjs — плюсы:
+ Понравилось, насколько просто в sails создается api
+ Очень классные возможности конфигурирования, начиная от шаблонного движка, БД, и используемого ORM (планирую прикрутить bookshelfjs.org/ на sails)
+ Очень понравилось, что есть готовые grunt таски, которые неплохо решают задачу генерации как прод так и дев бандлов.
+ есть команда (sails www) которая собирает только клиенский код — удобно для отделения работы фронта и сервера.
+ поддержка мультиязычности (не юзал, но знаю, что есть)

Минусы:
— на данный момент багнутая работа model assocoations (понимаю, что v0.10 — еще бета, а в v0.9.x — этого вообще нет)
— поддержка лейоутов только для ejs шаблонов

Ractivejs — плюсы:
+ возможность привязки к backbone
+ расширяемость (можно писать свои плагины)
+ удобный шаблонизатор на основе mustache (не люблю ejs — очень громоздкий как по мне)
+ хорошая дока, примеры и туториал

Ractive — минусы за несколько дней использования не обнаружил.

Благодарю за внимание.

sails.js, backbone.js, ractive.js

комментарии (0)

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.

Стрелочные функции (Arrow functions) в ECMAScript 6 | Стимпанк-кофейня в Южной Африке | 8 ловушек программирования

Войти
Регистрация

Разделы
Хабы
События
Компании
Пользователи

Посты
Лучшие
Тематические
Корпоративные
Песочница

Инфо
О сайте
Правила
Помощь
Соглашение

Услуги
Реклама
Корпоративные тарифы
Семинары
Спецпроекты
Льготы стартапам

ТМ © 2006-2014
Служба поддержки
Мобильная версия
Мобильные приложения

Brainstorage — Все мозги в одном месте
ТОСТЕР — Q&A-сервис для разработчиков
ФРИЛАНСИМ — Заказы для фрилансеров
АВТОКАДАБРА — Уютная и дружелюбная