Bricks Pro. Developer’s Guide.

Ver. 1.0 от 3.06.2017.

Разговорный стиль. Текст частично скопирован из чата.

Здоровый самотроллинг. Автор признаёт свою несостоятельность в некоторых моментах, что выражается в виде шуток над самим собой и над своими решениями.

Краткая история создания фреймворка.

Когда я пытался изучать язык php в 2007 году, мне попался клиент, которому надо было переписать сайт. Тогда я за пару недель написал простейший движок для сайта, который нельзя было назвать фреймворком, но на нём можно было сделать сайт. В те времена у меня зародилась идея, что я могу сам для себя сделать удобный инструмент для разработки софта под интернет, и поэтому я не жалел времени и сил на развитие этой идеи. От проекта к проекту, мой код улучшался, добавлялись какие-то возможности.

Но этот движок был весьма сложен в масштабировании. Сделать на нём сайт размером более 10 страниц было непростой задачей. Я стал чётко видеть серьёзные недостатки, которые мешали работать всё сильнее и сильнее...

Однажды в августе 2012 года я пытался устроиться на работу удалённо программистом и меня спросили, знаю ли я, что такое паттерна MVC. Я быстро загуглил сабж, прочитал пару строк в википедии, нашёл интересную статью на хабре и ответил работодателю, что да, знаю. В ответ он меня попросил написать в качестве тестового задания небольшое веб-приложение с твиттероподобным функционалом, что я и сделал. Я изучил ту статью подробно, и, благо в ней были примеры кода, быстро набросал свой движок. Так родился фреймворк Bricks, он стал развиваться, я начал делать на нём разные интересные приложения и вскоре переименовал его в Bricks Pro, обнаружив, что у меня теперь есть очень удобный, гибкий, быстрый в разработке и универсальный инструмент для создания чего угодно.

Любопытно, что некоторые последствия моих экспериментов десятилетней давности используются и по сей день в виде функций и библиотек в Bricks Pro.

Особенности фреймворка:

- Поддержка HTML-шаблонов (переменные, циклы, разграничение прав доступа, элементы управления — в виде псевдотегов HTML)

- Простота интеграции со сторонними сервисами
- Паттерна MVC OOP на бэкэнде, php 5.x
- Простота масштабирования (по мощности, по скорости разработки)
- Возможность использования совместно с frontend-фреймворками, например bootstrap
- Поддержка расширенных таблиц стилей less с компиляцией на стороне сервера
- Широко используемый динамический обмен данными (ajax)

Устройство фреймворка

Точка входа для всех запросов — файл index.php.

Как правило, в корне кроме index.php находится также файл settings.php, в котором содержатся различные конфигурационные данные (такие, как доступ к базе данных, название проекта, пути к разным группам файлов, доступы к сторонним сервисам и прочие данные).

Кроме того, в корне находится, как правило, ещё 4 папки — controller, data, model, view. Рассмотрим их назначение подробнее.

controller

На данный момент (июнь 2017 г.) существует две разновидности фреймворка — для веб-приложений и для сайтов. Второй отличается тем, что в нём 4 контроллера вместо одного. В версии для веб-приложений физически контроллер размещён в одном файле, но по структуре логически его можно условно разделить на те же 4 части (что и было сделано однажды перед созданием очередного сайта на Bricks Pro, поняв, что размер файла Controller.php превзошёл все мыслимые и немыслимые границы).

Короче говоря, независимо от разновидности фреймворка, контроллер состоит из четырёх следующих частей:

- **Роутер** — направляет запросы туда, куда надо, учитывая авторизацию пользователя.
- **Файловый** — обрабатывает загрузку файлов.
- **Ajax** – обрабатывает XHR-запросы. Action обрабатывается через select.
- **HTTP** – обрабатывает загрузку страниц.

Кроме того, на контроллер возложены следующие функции:

- Заполнение шаблонов (класс Template).
- Минификация CSS и JS кода (срабатывает автоматически при обновлении соответствующих файлов).
- Компиляция файлов LESS (срабатывает автоматически при обновлении соответствующих файлов).
- Логирование запросов к серверу и некоторых событий (как правило, в файловый кеш).
- Работа с сессиями (используется собственный обработчик сессий SessionSaveHandler).

- Обработка строковых данных (StringForge), в том числе поддержка русских переносов в тексте, в случае необходимости.
- Локализация (через языковые псевдотеги в шаблонах и функцию localize()).

Также в этой папке есть подпапка js с кодом на языке JavaScript. Как правило, код js для фронтэнда админ-панели размещён в файле common.js. Также присутствует несколько библиотек:

- ajax2.js — для работы с ajax-запросами.
- json.js — для работы с json.
- md5.js – для работы с md5.
- … и прочие, такие, как jQuery.

Библиотеки для работы с ajax и json были созданы до того, как появился jQuery, и в связи с привычкой было решено не переходить на jQuery-запросы, а придерживаться старого стиля кодирования. Множество примеров вызова таких запросов можно найти в файле common.js. То же касается причины использования собственной библиотеки для работы с json.

> *Особенности работы ajax.* Для динамической подгрузки контента через ajax в модулях админ-панели используется функция showtable в common.js. Она автоматически собирает все фильтры в один json-объект и отправляет его на сервер. Во всех проектах на Bricks Pro для перезагрузки основного рабочего поля страницы (блок с id=xlspreview) используется клавиша Esc.

Возвращаясь к структуре подпапок, допускается нахождение в этой папке подпапок со сторонними библиотеками, которые хотя бы условно можно отнести к контроллеру (например, минификатор кода) — как php, так и js (в подпапке js).

data

Эта папка предназначена для хранения различных полезных (и не очень) файлов, которые не относятся ни к одной из частей паттерны MVC. В частности, в этой папке могут храниться разнообразные логи, файлы кеша, различные сообщения о запросах SQL, сессии пользователей, контент (фотографии, картинки, загруженные файлы), словари для минификатора и для расстановки переносов в тексте, промежуточные результаты обработки данных, и прочее.

model

Модели постигла та же участь, что и контроллер — основной файл Model.php по размерам превосходит даже Controller.php. В относительно свежих проектах (начиная с 1 марта 2017 г.) серверная логика группируется в трейты (traits) и сохраняется в файлы с именами вида Model_Ads.php. Соответственно, внутри таких файлов трейты называются, в данном примере, так:

```
trait Model_Ads {}
```

Исторически сложилось, что к модели также относится парсер настроек (Settings.php – здесь файл парсера называется так же, как сам файл настроек). Кроме того, к модели относится класс Database, который представляет собой оболочку для PDO, поддерживающую MySQL и postgresql, причём если вы не знаете особенности диалекта postgre – с вероятностью в 90% ваши запросы на диалекте MySQL будут правильно работать с базой данных

postgre. В остальных 10% случаев придётся посмотреть, каким должен быть запрос, чтобы он не вызывал ошибку при исполнении.

Также в этой папке, как правило, размещаются различные сторонние библиотеки, которые логически можно отнести к модели — например, различные SDK для интеграции с другими сервисами (FB, например).

Отдельного упоминания заслуживает файл legacy.php, который по сути является свалкой различного кода, который не вошёл ни в один из классов и ~~окончательно ставит жирный крест на инкапсуляции~~ значительно экономит время, когда надо добавить какую-то важную функцию, которую можно вызвать откуда угодно (особенно если мы ещё не знаем, откуда она будет вызываться). Человеку, который когда-нибудь разберёт этот файл и раскидает функции по классам, гарантируется +1 в карму от каждого, кто будет работать с этим фреймворком в будущем.

view

В этой папке может быть несколько подпапок, которые представляют собой темы оформления (пакеты шаблонов). В случае веб-приложения здесь будет единственная папка modern (это название тоже имеет историческое происхождение и обозначает название темы. Ранее была ещё одна тема classic, но она исчерпала себя и сейчас не поддерживается). В modern хранится админ-панель веб-приложения, и, в некоторых случаях, клиентская часть для неавторизованных пользователей. В случае веб-сайта здесь также будет другая папка portal, в которой хранится фронтэнд для сайта.

Каждая тема состоит из шрифтов (fonts, необязательная папка), графических файлов (img), стилей (styles. Основные стили хранятся в формате LESS и компилируются автоматически на стороне сервера, также возможно подключение статических CSS-файлов), файлов htt (templates. hypertext template – это тот же HTML, но с псевдотегами Bricks Pro).

Следует помнить несколько важных моментов.

style.less — это, как правило, файл стилей для админ-панели. **mainparent.htt** — основной файл фронтэнда для админ-панели, в случае веб-приложения. Папка view более, чем другие папки, часто изобилует лишними файлами, которые могут сбивать с толку не только своими названиями, но и содержимым. Поэтому следует быть внимательным и смотреть на даты изменения этих файлов. Часто их следует удалять (но держать резервные копии, чтобы случайно не удалить нужный код), тем самым облегчая жизнь всем, кто будет работать над этим проектом в будущем.

Иногда бывают сложности с обновлением файлов на клиенте из-за их кеширования на стороне клиента, для этого используется загрузчик фронтэнда file.php, находящийся в корне фреймворка. Он выдаёт на фронтэнд различные (как правило, статические) файлы с правильными заголовками. Пример использования можно подсмотреть в mainparent.htt.

> *Лирическое отступление.* Вообще, теоретически можно разместить весь фронтэнд в одном единственном файле .htt, но так не сделано исходя из соображений логической организации кода и быстродействия.

Для упрощения работы с однотипными фрагментами кода в шаблонизаторе реализован так называемый механизм Common Controls, который представляет собой аналог элементов управления в визуальных средах разработки вроде Visual Studio, Delphi и им подобных. Суть в том, что часто повторяющиеся фрагменты разметки HTML помещаются в специальные шаблоны с именами файлов вида block_calendar.htt, и впоследствии вставляются в шаблоны в

виде псевдотегов с параметрами (которые разделяются пробелами). Примеры использования Common Controls можно часто встретить в файлах .htt.

Политика именования файлов (здесь вместо звёздочки, как правило, бывает название модуля, которое передаётся в GET-параметре «go», но в некоторых частных случаях могут быть вариации на эту тему):

editor_*.htt – это файлы всплывающего редактора записи (карточки). Иногда у одного модуля бывает несколько таких редакторов.

left_*.htt – это файлы левой панели фильтров. Как правило, содержат в себе собственно набор фильтров, поле статуса (status0) и шаблон всплывающего окна редактора.

main_*.htt – это файлы шаблона основного рабочего поля модуля, как правило представляют собой таблицу с множеством полей. Столбцы могут быть прописаны статически в шаблоне, либо запрограммированы через цикл в контроллере.

Кстати, про циклы. Конструкция вида

```
<select class=filteritem id=flt_Month>
  <option id="0"><ru>Все</ru><en>All</en></option>
  <loop_months>
    <option id="%Year%-%Month%" %selected%>%Year%-%Month%</option>
  </loop_months>
</select>
```

представляет собой типичный цикл для шаблонизатора Bricks Pro. В этом примере через цикл реализовано заполнение опций в теге select. Такая конструкция очень часто используется в фильтрах (left_) и в карточках (editor_). Тот же функционал циклов используется для подгрузки отдельных блоков кода:

```
<block_no>
  <en>No</en><ru>Нет</ru>
</block_no>
```

такая конструкция иногда используется для замены в контроллере нулей на слово «Нет», а также в некоторых других случаях (например, для украшения иконками). Исторически она возникла в проекте базы данных для риелторов, где при отсутствии балкона или лоджии нужно было явно указывать их отсутствие вместо вывода цифры «0».

Циклы поддерживают бесконечную вложенность друг в друга. Код на php я не привожу в связи с тем, что его можно подсмотреть в контроллере самостоятельно.

Другая обязательная составляющая любого шаблонизатора — это переменные. В Bricks Pro они заключаются в знаки %:

```
<td>%DateAdded%</td>
```

Циклы, переменные и элементы управления обрабатывает класс Template.

Интернационализация (она же локализация). Весь фронтэнд поддерживает локализацию через псевдотеги языков. Например, ранее встречалась такая конструкция:

```
<en>No</en><ru>Нет</ru>
```

Нужный язык хранится в сессии, таким образом, легко переключиться на другой язык из фронтэнда, передав в get-параметре значение переменной lang, например, lang=en.

Допустимые языки и язык по умолчанию хранятся в коде в файле model/localizaion.1.x.php:

```
 function localize($buf) {

  $langs = array("ru", "en");   // list of available languages

  $lang='ru';                   // default language

  ...
```

Кстати, эта функция принимает на вход нелокализованный код HTML, и убирает из него всё, что находится внутри тегов ненужных нам языков, оставляя только текст на нужном нам языке. Теоретически, этот механизм позволяет работать с любым количеством языков, на практике же при создании приложений на Bricks Pro хорошим тоном считается создание шаблонов сразу на английском и русском языках.

Если мы имеем дело с базой данных, откуда строки выводятся на фронтэнд, то там тоже допускается локализация описанными выше средствами, включая правило хорошего тона.

Выше я упоминал про минификатор кода, в данном случае я имел ввиду js и css код. Кроме минификации, производится склеивание нескольких файлов в один, с целью уменьшения количества запросов к серверу и ускорения загрузки страниц. Для этого минификатор находит последовательность ссылок на такие файлы

```
 <script  type="text/javascript"  src="file.php?name=controller/js/ajax2.js"></
script>

 <script  type="text/javascript"  src="file.php?name=controller/js/common.js"></
script>

 <script type="text/javascript" src="controller/js/dhtmlxca.js"></script>

 <script type="text/javascript" src="controller/js/json.js"></script>

 <script type="text/javascript" src="controller/js/md5.js"></script>


 <script     type="text/javascript"  src="controller/js/jquery-1.8.3.min.js"></
script>
```

и собирает их в одну

```
 <script  type="text/javascript"  async  defer  src="file.php?name=controller/js/
mainparent.js"></script>

 <script     type="text/javascript"  src="controller/js/jquery-1.8.3.min.js"></
script>
```

Здесь видим, что последняя ссылка не прикрепилась. Это сделано специально, потому как 1) код библиотеки jQuery здесь уже сжатый и 2) на практике оказалось, что минификатор ломает эту библиотеку и она перестаёт работать правильно (начиная работать неправильно). Это произошло благодаря двум пробелам между словами script и type.

Похожим образом происходит обработка файлов css.

Из вышесказанного делаем вывод: никогда не редактируйте файл mainparent.js (и style.css тоже, по той же причине).

Заключение

В планах на будущее — разделение кода на core и application, что позволило бы обновлять ядро на всех проектах сразу. В остальном этот фреймворк хорошо себя зарекомендовал как инструмент быстрого создания действующих прототипов веб-приложений, и в большинстве случаев быстродействия и функционала оказывалось достаточно для полноценной работы приложений на его основе. В связи с этим считаю, что Bricks Pro следует продолжать использовать для новых проектов, улучшая его архитектуру и ходовые качества.