

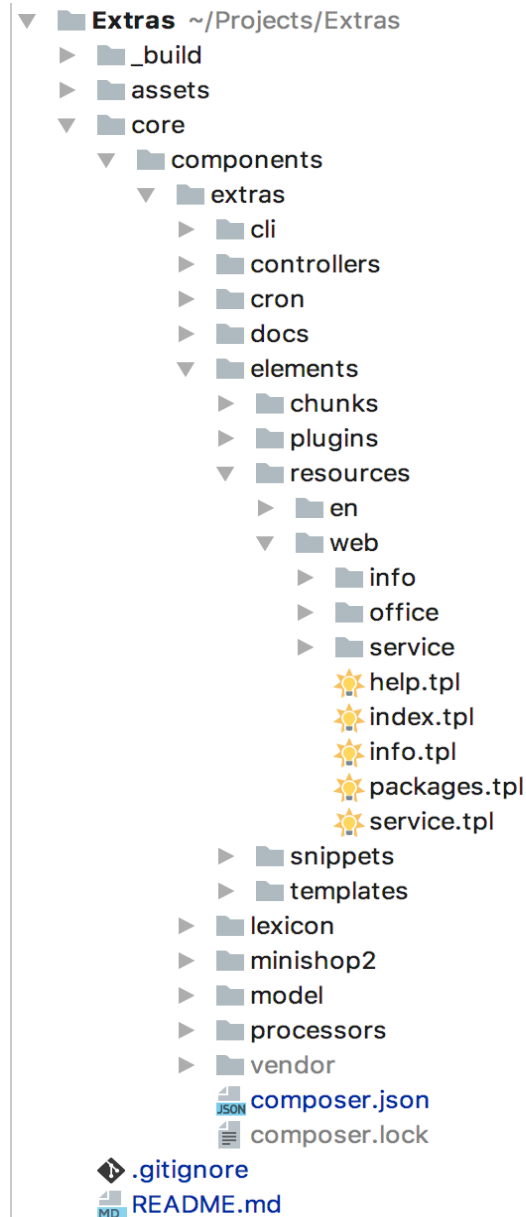
Update version modstore.pro

Primary targets

- Updating page proofs, input adaptability
- English version of the store
- Improvement of personal account, particularly the key support and governance section
- Refactoring of repository code as he used the outdated modRestServer
- Carry of supervisory section on frontend

It uses its own transport packet **Extras** for development, which:

- Set all patterns, snippets, chunks and resources
- Supply / change system settings, file source
- Set its own order handler miniShop2 and controllers of user account
- Copy and update scripts, patterns and images
- Allow to work conveniently with all of these from IDE PhpStorm and allow to use the version-control



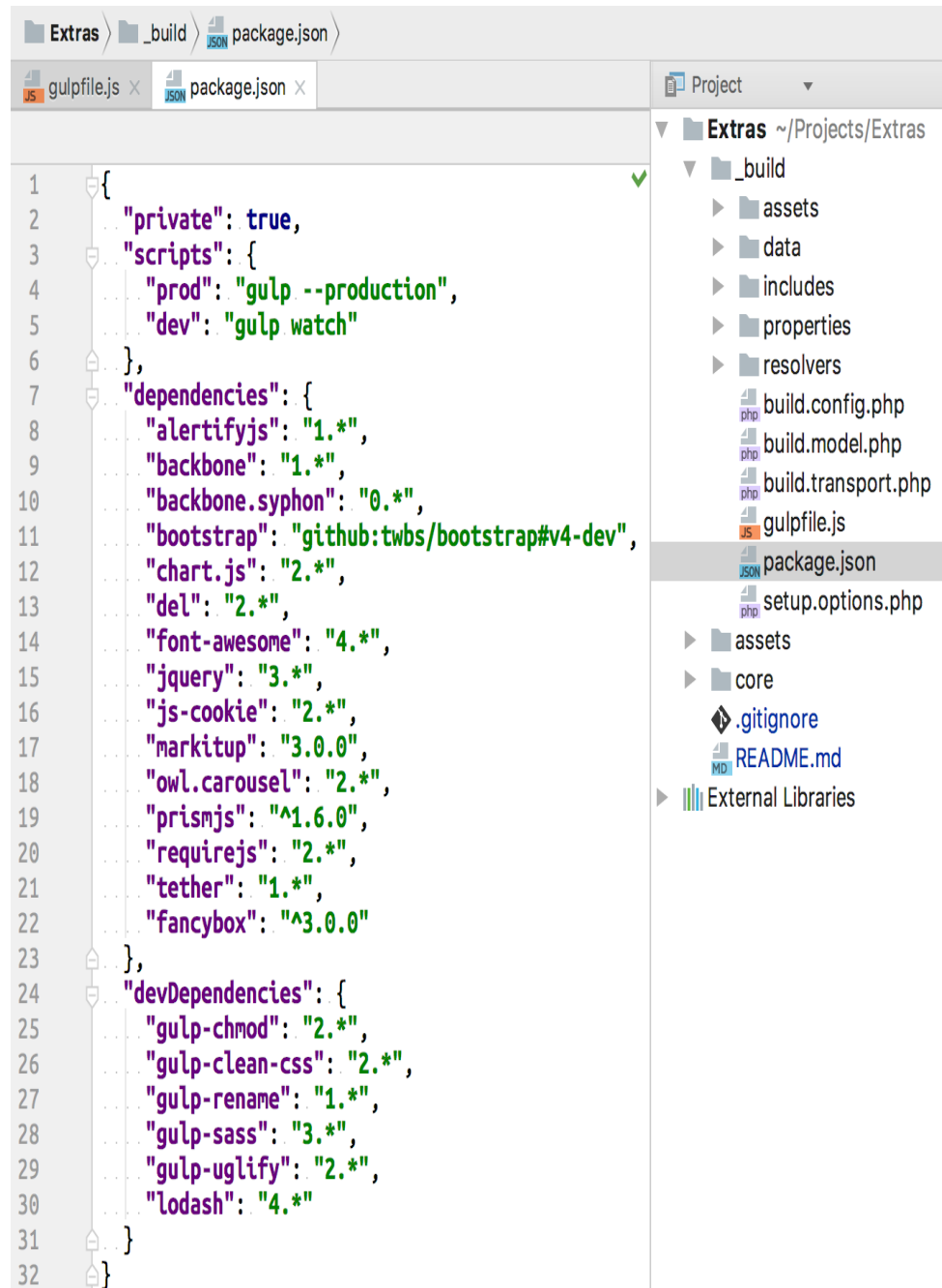
Implemented appendings

- miniShop2 + payment tools for store installation
- Office + HybridAuth for profile authorisation and reprogramming
- mSearch2 for searching and screening appendings
- Tickets for dividing supports
- pdoTools for supporting of work all these appendings and Fenom template engine

Frontend

- MinifyX complete rejection
- Files integration and minification via Gulp
- Using RequireJS for dividing scripts logic onto small modules

```
requirejs(["app/orders"],  
    function(App) {  
        App.Orders.init()  
    }  
);
```



The screenshot shows an IDE window with a file explorer on the right and a code editor on the left. The file explorer shows a project structure with folders like _build, assets, data, includes, properties, resolvers, and files like build.config.php, build.model.php, build.transport.php, gulpfile.js, package.json, and setup.options.php. The code editor shows the content of package.json, which includes private, scripts, dependencies, and devDependencies sections.

```
1 {  
2   "private": true,  
3   "scripts": {  
4     "prod": "gulp --production",  
5     "dev": "gulp watch"  
6   },  
7   "dependencies": {  
8     "alertifyjs": "1.*",  
9     "backbone": "1.*",  
10    "backbone.syphon": "0.*",  
11    "bootstrap": "github:twbs/bootstrap#v4-dev",  
12    "chart.js": "2.*",  
13    "del": "2.*",  
14    "font-awesome": "4.*",  
15    "jquery": "3.*",  
16    "js-cookie": "2.*",  
17    "markitup": "3.0.0",  
18    "owl.carousel": "2.*",  
19    "prismjs": "^1.6.0",  
20    "requirejs": "2.*",  
21    "tether": "1.*",  
22    "fancybox": "^3.0.0"  
23  },  
24  "devDependencies": {  
25    "gulp-chmod": "2.*",  
26    "gulp-clean-css": "2.*",  
27    "gulp-rename": "1.*",  
28    "gulp-sass": "3.*",  
29    "gulp-uglify": "2.*",  
30    "lodash": "4.*"  
31  }  
32 }
```

- The scripts of appendings are copied and minified by separate command of Gulp, in order to connect them as a part of main application
- Own initialization of scripts by appendings is **deactivated**, no of js at the head of page
- If required – these scripts are driven from the main application, by preliminary changing operation

```
requirejs.config({
  baseUrl: '/assets/components/extras/js/web/',
  urlArgs: 'v=' + document.head.querySelector('meta[name="assets-version"]').content,
  waitSeconds: 30,
  paths: {
    jquery: 'lib/jquery.min',
    bootstrap: 'lib/bootstrap.min',
    tether: 'lib/tether.req',
    backbone: 'lib/backbone.min',
    backbone_syphon: 'lib/backbone.syphon.min',
    underscore: 'lib/underscore.min',
    alertify: 'lib/alertify.min',
    jquery_owl: '/assets/components/extras/js/web/lib/owl.carousel.min',
    cookies: '/assets/components/extras/js/web/lib/js.cookie.min',
    prism: '/assets/components/extras/js/web/lib/prism.min',
    markitup: '/assets/components/extras/js/web/lib/markitup.min',
    chart: '/assets/components/extras/js/web/lib/chart.min',
    fancybox: '/assets/components/extras/js/web/lib/jquery.fancybox.min',
    minishop2: '/assets/components/extras/js/web/lib/minishop2.min',
    msearch2: '/assets/components/extras/js/web/lib/msearch2.min',
    jqui: '/assets/components/extras/js/web/lib/jquery-ui.min',
    pdopage: '/assets/components/extras/js/web/lib/pdopage.min',
  },
  shim: {
    bootstrap: {
      deps: ['jquery', 'tether']
    },
    underscore: {
      exports: '_'
    },
    backbone: {
      deps: ['underscore', 'jquery'],
      exports: 'Backbone'
    },
    alertify: {
      exports: 'alertify'
    },
    minishop2: {
      deps: ['jquery'],
      exports: 'miniShop2'
    },
    msearch2: {
      deps: ['jquery'],
      exports: 'mSearch2'
    }
  }
});
```

For example, miniShop2 initialization:

```
initMiniShop2: function () {
  miniShop2.Message = App.utils.Message;

  miniShop2.Cart.status = function (status) {
    var $miniCart = $(miniShop2.Cart.miniCart);
    if (status['total_count'] < 1) {
      $(App.Order.elem).modal('hide');
      $miniCart.removeClass(miniShop2.Cart.miniCartNotEmptyClass);
    }
    else {
      if (status['total_count'] > 0 && !$miniCart.hasClass(miniShop2.Cart.miniCartNotEmptyClass)){
        $miniCart.addClass(miniShop2.Cart.miniCartNotEmptyClass);
      }
      $(miniShop2.Cart.totalCount).text(status['total_count']);
      if ($(miniShop2.Order.orderCost, miniShop2.Order.order).length) {
        miniShop2.Order.getcost();
      }
    }
  };
};
```

**** feature updating does not break anything***

From this perspective

- Pages load as much quickly as possible



Для мобильных



Для компьютеров

95 / 100 Рекомендации

- Redundant code is not run
- Total control over javascript logic work of external appendings – since having the possibility to change almost every operation

So, these modules look like:

```
<script type="text/javascript">requirejs(["app", "app/office"]);</script>
  <div class="modal fade" id="order">
    <div class="modal-dialog">
      <div class="modal-content">
        <div class="modal-header">
          <h4 class="modal-title">Загрузка...</h4>
          <button type="button" class="close" data-dismiss="modal">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          <i class="fa fa-cog fa-spin"></i>
        </div>
      </div>
    </div>
  </div>
```

```
<script type="text/javascript">requirejs(["app", "app/order"]);</script>
<script type="text/javascript">requirejs(["app", "app/remote-comments"], function(App) {App.RemoteComments()});</script>
<script type="text/javascript">requirejs(["app", "app/index"]);</script>
</body>
</html>
```

**** every script is loaded only once***

English version

Items

- 2 scopes, for each case its own items miniShop2, connected with repository packets through *package* field
- Originators can edit them separately in user account, switching on or off as an option
- Extended class of a order checks keys and buying appendings at the time of the order

The screenshot displays the 'pdoTools' web interface. At the top, there are two buttons: 'Сохранить' (Save) and 'Копировать' (Copy). Below these are navigation tabs: 'Товар' (Item), 'Группы ресурсов' (Resource groups), and 'Комментарии' (Comments). Underneath, there are sub-tabs: 'Документ' (Document), 'Настройки' (Settings), 'Свойства товара' (Item properties), 'Связи' (Links), and 'Категории' (Categories). The 'Свойства товара' tab is active, showing several input fields: 'Цена:' (Price) with an empty text box, 'Старая цена:' (Old price) with a text box containing '0', 'Дополнение:' (Addition) with a dropdown menu showing 'pdoTools' and a list of other items including 'yTranslit', 'YandexMaps', 'YaCaptcha', 'xPoller2', 'xParser', and 'xButtons', and 'Комиссия магазина:' (Store commission) with a text box containing '0'. At the bottom right, there are two checkboxes: 'Новый' (New) which is unchecked, and 'Особый' (Special) which is checked.

Lexicons are fairly well, but inconvenient

- There is a necessity to write them separately in chunks and wordindexes
- There is a necessity to track on internal logic of naming and remember which lines are already used
- Finally, the most important – there is a need to clean cache memory after nodding or changing the text, which makes a work via IDE complex

For these reasons a decision was taken to use lexicons only within PHP and Javascript.

Accessor \$.en

Fenom allows to add operations from plugins on signal pdoToolsOnFenomInit:

```
$fenom->addAccessorSmart('en', 'en', Fenom::ACCESSOR_PROPERTY);
```

After that they can be used in formatting:

```
{if $.en}  
    Long english text  
{else}
```

Long Russian text

```
{/if}
```

Letter formalisation

- All letters are inherited from one pattern via Fenom
- For convenient tracking is used CSS classes

```
<style>
```

```
body > table { width: 600px; margin: auto; }
```

```
p { font-size: 16px; line-height: 22px; }
```

```
</style>
```

The point is that Gmail does not follow those styles – it requires to shape everything inline:

```
<p style="font-size: 16px; line-height: 22px;">
```

```
My paragraph
```

```
</p>
```

Packagist has striking library for integration styles inline into appropriate tags - pelago/emogrifier.

But how to make it prepare **all** site letters?

It has emerged that very simple:

```
$modx->getService('mail', 'extraMail',  
    MODX_CORE_PATH . 'components/extras/model/'  
);
```

The trick is to do it foremost, for instance, by plugin OnMODXInit, after that `modx::getService()` will always load our extended class of working with mail.

Further on mere formality:

```
class extraMail extends modPHPMailer
{
    public function set($key, $value)
    {
        if ($key == modMail::MAIL_BODY) {
            // This is it – the integration styles into tags
            $emogrifier = new \Pelago\Emogrifier($value);
            $value = $emogrifier->emogrify();
        }

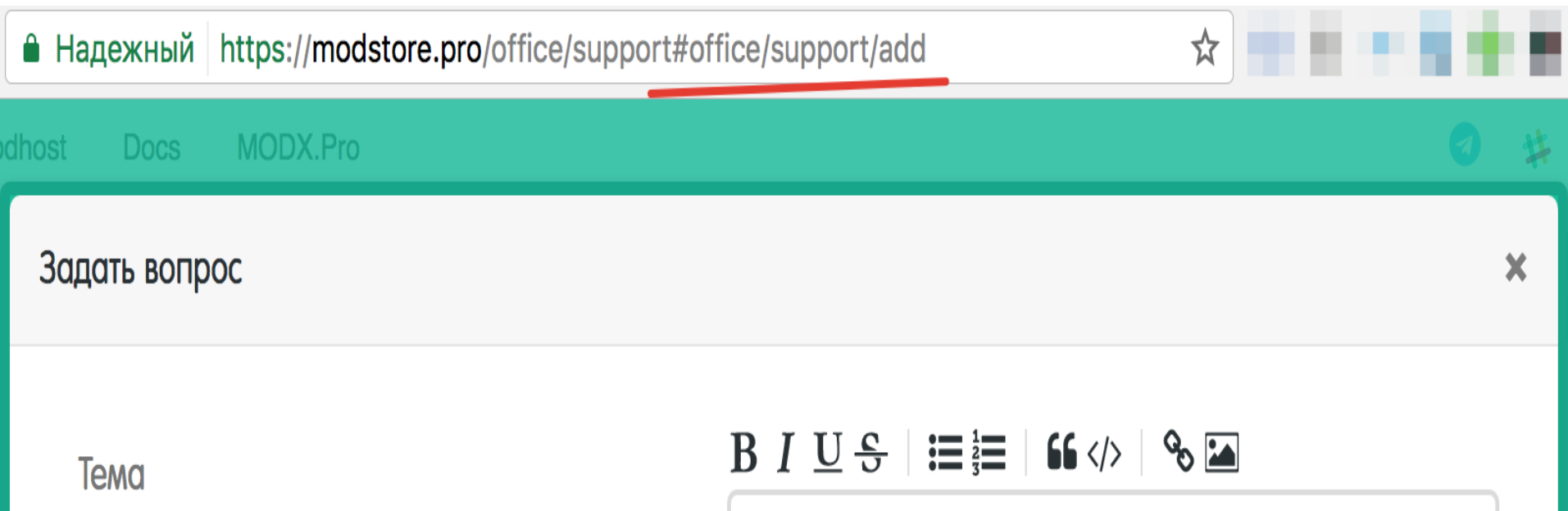
        parent::set($key, $value);
    }
}
```

So now we can easily shape letters, not writing manually styles into every tag.

User account

Main features:

- Single connector across inquiries
- Checking CSRF token
- Direct links for all actions



Exceptional features

- Capability to change the mode of payment for new orders
- Deleting the unpaid offers
- e-mail notification of originators about purchasing their appendings
- Favourites and attaching into basket from it, selectively a key
- Technical support with new version MarkItUp and syntax highlighting Prism

Thank you!