Инструкция по эксплуатации, настройке и установке

ПРОГРАММЫ ДЛЯ ЭВМ

Веб-фреймворк «БитПлатформа»

ООО «Митлабс»

2020

Оглавление

1. Технические требования

1.1. Перечень необходимого программного обеспечения:

Операционная система: операционная система под в рамках которой возможно функционирование нижеперечисленных программ с указанными настройками:

a) Apache HTTP Server не ниже 4-й версии с включёнными модулями:

- mod_rewrite
- mod_mime
- mod_deflate
- mod_expires

b) PHP Версия: 5.3 и выше с включенными модулями:

- curl,
- iconv,
- json,
- mbstring,
- gd2,
- mcrypt,
- pdo,
- pdo_mysql,
- xml

Настройки PHP:

- error_reporting E_ALL | E_STRICT
- log_errors On
- php_flag display_errors On
- register_globals Off
- magic_quotes_gpc Off
- magic_quotes_runtime Off
- short_open_tag Off
- date.timezone Europe/Moscow

c) MySQL 5.0.77-4 и выше (поддерживается СУБД MySQL Community Edition)

d) NGINX (опционально)

Модули NGINX (в случае использования)

- ngx_http_gzip_module
- ngx_http_gzip_static_module
- ngx_http_headers_module

1.2. Требования к оборудованию:

- 64-разрядный процессор с тактовой частотой не ниже 1,4 ГГц, совместимый с набором инструкций для архитектуры x64.
- 8 Гб ОЗУ,
- 500Мб ПЗУ
- Сетевое оборудование подключённое к компьютерной сети

2. Система управления содержимым сайта

Сайт представляет собой совокупность разделов, которые связаны общими элементами навигации. Каждый раздел сайта состоит из набора HTML-страниц, которые могут быть как статическими (т.е. создаваемыми и обновляемыми вручную), так и динамическими, т.е. генерируемыми специальными программными модулями. Обновление динамических страниц производится автоматически. Содержание HTML-страниц сайта редактируется в административном разделе (back-office сайта).

Программные модули при генерации страниц используют информацию из базы данных (БД) сайта. Для отдельных динамических страниц (страниц информационных разделов) такой информацией могут быть дополнительные изображения, файлы, загружаемые для отдельных элементов сайта, например, товара в каталоге, связанные с ним элементы и т.п. Число элементов информационного раздела (статей, новостей, товаров в каталоге) не ограничено. В настройках сайта определяется максимальное число элементов на странице, программно задается принцип их сортировки по умолчанию (описывается отдельно для каждого раздела), механизм дополнительной сортировки в публичной части определяется дизайном. В случае превышения числа элементов над максимальным числом элементов на странице, реализуется автоматическое разбиение элементов на страницы и возникает постраничная навигация внутри раздела.

Управление информацией в БД (добавление, модификация, удаление) осуществляется посредством административных программных модулей.

Back-office сайта позволяет администратору управлять структурой сайта, содержанием разделов и ресурсами сайта.

Содержимое информационных статических страниц и страниц информационных разделов редактируется с помощью визуального редактора. Визуальный редактор позволяет форматировать текст, создавая абзацы, отступы, списки, гиперссылки, текст выделенный курсивом, жирным, подчеркиванием, дает возможность создавать заголовки внутри текста. Визуальный редактор позволяет добавлять в текст изображения, настраивая положение изображения внутри текста, отступы, обтекание текстом. Также администратор может создавать в визуальном редакторе таблицы.

Визуальный редактор применяет к тексту так называемый «язык разметки», указывающий браузеру пользователя, какая часть текста, например, является заголовком, а какая — пунктом списка. При отображении содержимого информационных статических

страниц и страниц информационных разделов - к ним применяются стили дизайна, определяющие цвета, отступы между элементами, называния и группы шрифтов и т.д. Таким образом, все тексты, в том числе отформатированные в визуальном редакторе, отображаются на сайте в едином стиле.

Для управления содержимым на сервер устанавливается и настраивается Bitplatform CMS (Content Management System или Система управления содержимым, административная часть). Система управления даёт возможность изменять содержимое сайта, хранящееся в базе данных, добавлять и удалять страницы. Шаблоны, графические элементы, являющиеся частью графического оформления внешней части сайта, анимационные элементы, формы и другое содержимое не внесённого ранее через систему управления содержимым, главная страница и некоторые другие автоматически генерируемые страницы, описанные в разделе «Описание разделов сайта, страниц и сервисов», не могут быть изменены через систему управления.

Внешний вид (дизайн) административной части сайта, расположение элементов управления определяются Исполнителем. По желанию Заказчика могут быть изменены основные (зелёные) цвета административной части на фирменные, а также заменён логотип BitPlatform на логотип компании Заказчика. В этом случае логотип BitPlatform перемещается в нижнюю часть страницы управления под горизонтальную полосу.

Настройки конфигурации сервера и управление серверными директивами и серверным программным обеспечением выходит за рамки функциональности системы управлением содержимым.

Интерактивные элементы управления содержимым (поля для ввода, визуальные редакторы, загрузчики файлов) позволяют изменять содержимое страниц сайта в рамках описанной в данном техническом задании функциональности; предоставляются в неизменяемом виде, соответствующем изложенным в техническом задании требованиям.

Система администрирования проекта устанавливается на веб-сервер, т.о. становится доступной с любого компьютера подключённого к интернет. При вводе логина и пароля для доступа в систему пользователь получает полные права для создания, редактирования и удаления содержимого.

Удаление информации, осуществляемое из администраторской панели, происходит безвозвратно. Резервное копирование данных системой управления не производится и должно предприниматься компанией, занимающейся размещением сайта в Интернете, согласно требованием, указанным в разделе «Требования к серверу».

2.2. Управление страницами

Механизм управления структурой сайта и страницами реализуется четырьмя основными экранами:

1. список страниц

2. экран создания страницы
3. экран редактирования страницы
4. экран с корзиной страниц

Каждый из трёх экранов содержит дерево страниц сайта с возможностью перехода на экран редактирования страницы.

Экраны редактирования и создания страницы идентичны, однако на экране редактирования страницы нельзя менять тип страницы, если он уже был выбран и сохранён.

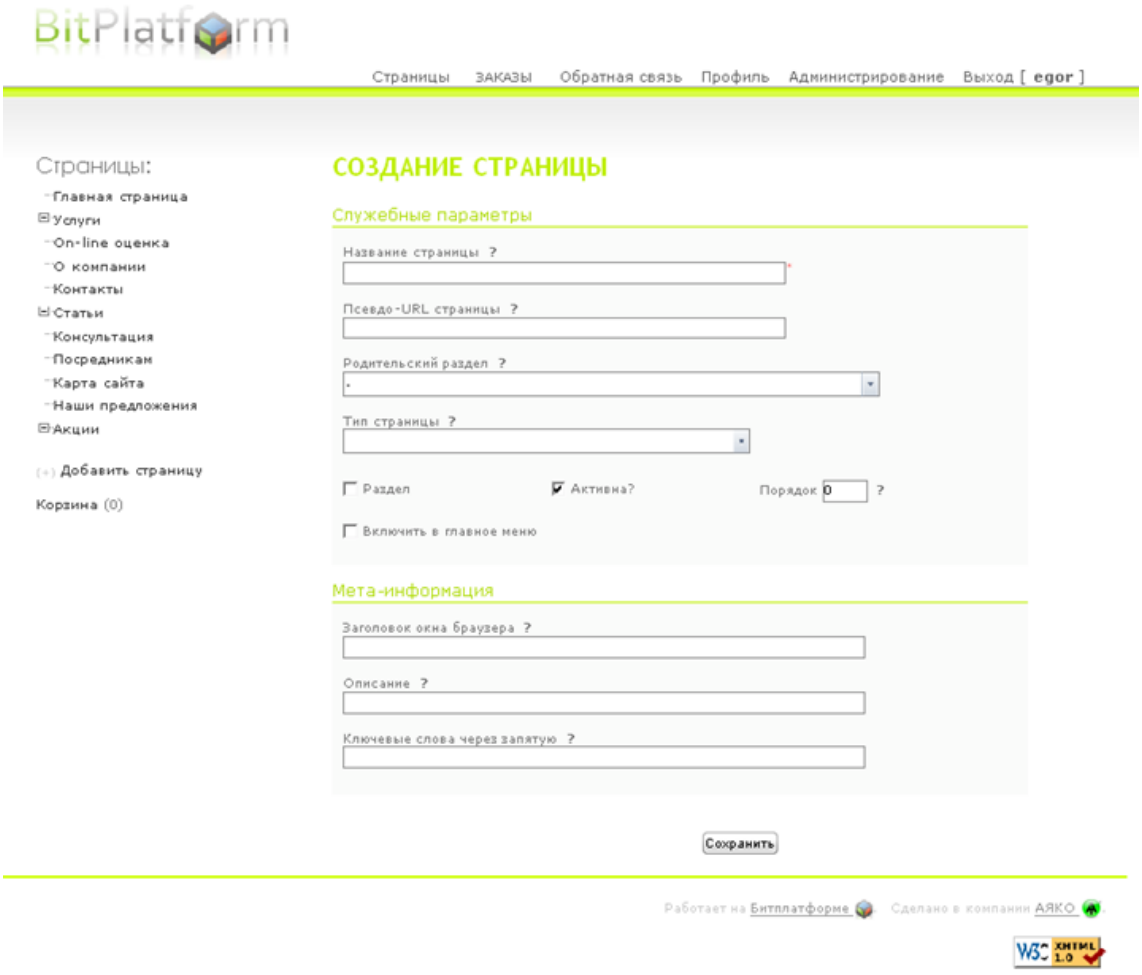

Рис. 1 Общий вид формы создания и редактирования страницы

Экраны редактирования и создания страницы содержат следующие обязательные поля:

1. Название страницы — текстовая строка, которая будет представлять страницу в списках и в дереве страниц;
2. Псевдо-URL страницы — идентификатор страницы, используемый для её получения через адресную строку браузера, не должен содержать кириллические и другие недопустимые в адресной строке символы;
3. Родительский раздел — Родительская страница, созданная ранее и помеченная как «Раздел»
4. Флаг «Раздел» — её необходимо отметить, чтобы страница могла стать

родительским разделом для других страниц;

5. Флаг активности страницы;
6. Порядок страницы — числовой коэффициент, от которого зависит позиция страницы в общих списках: чем выше коэффициент, тем выше страница;
7. Флаг включения в страницы в главное меню сайта;
8. Поле выбора типа страницы (активно только до момента выбора и сохранения типа страницы);
9. Поля для редактирования содержимого страницы в зависимости от её типа;
10. Поля для ввода мета-данных страницы: заголовка окна браузера, ключевых слов и описания.

Список базовых обязательных полей для всех страниц может быть пополнен в случае необходимости и не считается окончательным.

Созданную страницу можно переместить в корзину.

Экран корзины содержит дерево страниц и список удалённых страниц с возможностью удалить страницу безвозвратно или восстановить.

2.3. Как добавить тип страницы

Тип страниц описывает список полей, характерных для данного типа.

Для того, чтобы добавить новый тип страниц необходимо:

1. создать таблицы в БД со связями, соответствующими описываемым данным.

2. Добавить в таблицу greeny_pageTypes новую запись, где pageTypeID - порядковый номер записи (автоматический), name - название таблицы данных, description - описание, constantName - имя константы для связи данных, принято использовать ИМЯ_ТАБЛИЦЫ_ID.

3. Добавить все поля созданной таблицы с данными нового типа в таблицу greeny_fieldsRegistry используя системную утилиту /admin/fieldsregistry/. Вводим имя таблицы и нажимаем кнопку 'Сгенерировать форму'. Форма содержит ледующие поля:

- fieldName – имя поля таблицы
- controlType – тип элемента управления для поля
- description – текстовое описание поля
- canChange – флаг, разрешающий изменение поля
- visible – флаг, разрешающий отображение соответствующего элемента управления
- isListingKey - ?
- isMultiLanguage - ?
- tip – всплывающая подсказка, отображаемая при наведении мыши на элемент управления
- controlSettings – настройки элемента упр-я (в виде массива), позволяют задать такие параметры, как ширина, высота, порядок элемента на странице, связь с внешними таблицами и т. д.

После выполнения 3-х пунктов в списке типов страниц добавится новый пункт с новым

типом, а при его выборе откроется блок данных с полями этого типа.

2.4. Типы элементов управления

Список существующих типов элементов управления в bitplatform на данный момент:

- LineInput – простое текстовое поле
- HiddenInput — скрытое поле, генерирует HTML тег <input type=”hidden” />
- RichText – визуальны редактор (FCKEditor)
- TextArea – поле ввода текста
- SingleSelect – выпадающий список с возможностью выбора одного значения. Может брать значения из поля enum или внешней таблицы
- Radio – набор радио-кнопок, функционально похож на SingleSelect
- CheckBox – флажок, вкл./выкл.
- MultipleSelect – список с множественным выбором, подходит для отображения связи «многие ко многим»
- DateInput – поле ввода даты
- DateMaskedInput – поле ввода даты с возможностью указания маски
- SWFUpload – загрузчик файлов на основе flash-компонента SWFUploader
- DamnUpload – загрузчик файлов на основе Jquery-плагина DamnUploader

2.5. Требования к поисковой оптимизации

URL-адреса страниц

Для адресации страниц используется система человеко-понятных имён страниц, суть которой состоит в следующем. Страницы сайта находятся между собой в древовидном иерархическом отношении (являются узлами дерева страниц с пустым корнем). Каждая страница имеет имя-идентификатор, называемое алиасом, состоящее из латинских символов в нижнем регистре, цифр, знаков подчёркивания и тире. Каждая страница должна иметь уникальный алиас на её уровне иерархии. Алиасы первого уровня основных разделов сайта определяются разработчиками и не могут быть изменены. Алиасы страниц нижних уровней определяются контент-менеджерами согласно условий понятности и читаемости (английский эквивалент названия, транслитерация названия на русском). Адрес каждой страницы складывается из её алиаса и алиасов родительских страниц до первого уровня.

Примеры

алиас страницы контактов

contacts

адрес страницы контактов

http://site.ru/contacts/

Контент-страница каталога 3-го уровня с алиасом arbat, являющейся дочерней страницей раздела Москва (moscow), которая является дочерней по отношению к странице России(russia), являющейся дочкой страницы стран с алиасом countries.

адрес страницы

http://site.ru/countries/russia/moscow/arbat/

Мета-информация

Мета-информация – описание страницы, предназначенное для поисковых машин, включаемое непосредственно в каждую страницу с помощью мета-тегов. Система управления содержимым позволяет изменять текстовое содержимое всех страниц это допускающих через административную панель (за исключением тех, чьё содержимое генерируется автоматически, например, страница с результатом поиска или формой заявки), а также задавать для них содержимое мета-тегов и тега title. Главная страница допускает только изменение мета-тегов и тега title.

2.6. Требования к размеру и качеству загружаемых изображений и видео-файлов

Система управления содержимым позволяет контент-менеджеру загружать на сайт изображения и видео-файлы, однако они должны удовлетворять требованиям, приведённым в таблице ниже.

Описание требования	**Требование**
Формат загружаемых изображений	JPG, PNG, GIF (кроме прикреплённых к объектам недвижимости, т.е. только те, что загружаются через визуальный редактор)
Минимальная степень сжатия формата JPG	45%
Максимальный размер загружаемых изображений в пикселах	1024 по ширине и 724 по высоте
Разрешение	72 dpi
Максимальный размер в килобайтах	1024 килобайт

2.7. Раздел данных

Если нет необходимости представлять данные в виде страниц в виду частичного использования, смешивания с другими данными, использования данных на уже созданных страницах существует возможность оформлять данные таблиц в виде справочников.

Раздел данных представляет собой таблицу описанных в настройках (AnyEditorSettings.class.php) полей с возможностью поиска, фильтрации,сортировки, добавления, редактирования и удаления записей.

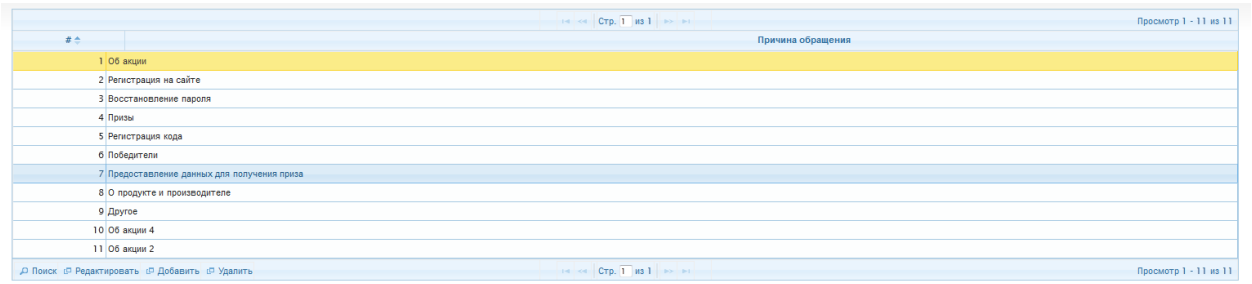

рис. 2. Общий вид таблицы радела данных

Страница редактирования записи - типичная форма, основанная на fieldsRegistry, т.е. включает в себя поля таблицы, описанные в greeny_fieldsRegistry и их вид соответствует выбранному типу элемента управления.

2.8. Как добавить раздел данных

1. Для добавления раздела данных, необходимо выполнить пункты 1 и 3 раздела «Как добавить тип страницы»

2. Отредактировать массив для данных в файле /applications/models/AnyEditorSettings.class.php. Каждый элемент представляет собой массив полей, описывающих раздел данных:

- ключ массива — название раздела данных (имя таблицы в нижнем регистре)
- tableName — имя таблицы
- ruName – русское название раздела
- ruItem – русское название записи в таблице
- listFields – массив полей, которые будут отображаться в таблице данных. Имеет следующий формат:
- ключ массива — название поля
- width – ширина колонки таблицы в пикселях
- sortable – флаг, разрешающий сортировку по данному полю
- searchable – флаг, разрешающий поиск по данному полю
- filter – условие, по которому будет проходить выборка записей (формат SQL WHERE)
- orderBy – условие начальной сортировки списка (формат SQL ORDER BY)

- limitOnPage – количество записей страницу
- allowAddition – разрешить добавление новых записей
- allowRemove – разрешить удаление существующих записей

2.9. Хлебные крошки

Хлебные крошки в CMS реализованы таким, чтобы обеспечить полную функциональность работы вне зависимости от предоставляемых данных (в виде страниц или данных).

Реализация содержится в *FrontController.class* через общую переменную *$breadcrumbs*:

```
protected $breadcrumbs = array();
```

Если нужно получить х/к для страницы, которая есть в БД, то ничего дописывать не нужно, они автоматически сформируются. Достаточно подключить *breadcrumbs.tpl* (по умолчанию подключен). Если такой страницы с алиасом нет или реализовано через данные, достаточно в общий массив добавить элемент:

```
$ar = array('name' => 'имя', 'alias' => 'алиас');

$this -> breadcrumbs[] = $ar;
```

И столько элементов, сколько необходимо для глубины х/к.

2.10. Пагинация в бек- и на фронт-енде

Пагинация в бек-енде и на фронт-енде реализована следующим образом (см. рис. 3).

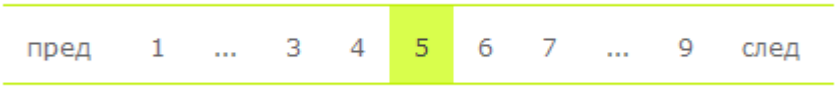

Рис. 3 - Пагинация

Максимальное количество элементов - 11, куда входит:

- 2-ве кнопки для перехода вперед и назад на одну страницу;

- 1-ая и последняя страница пагинации (для быстрого перехода);

- одна активная (текущая);

- 4-ри ближних страниц (по две с разных сторон от активной);

- 2-ва многоточия (позволяет перейти на предыдущую или следующую от последней ближней).

В зависимости от того какая страница активна и сколько осталось страниц элементы становятся доступными - см. рис. 4.

Рис. 4

2.11. Авторизация в CMS

Авторизация в CMS возможна также и на фронтенде, для этого в “админке” в разделе “Настройки” доступна опция - см. рис. 5:

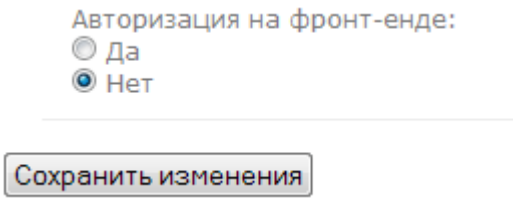

Рис. 5 - Авторизация на фронт-енде

Также по мимо “глобальной” авторизации, доступна “постраничная”, т.е. для каждой страницы отдельно. Для этого авторизация (логично предположить) на фронт-енде должна быть отключена. Чтобы сделать нужно в настройках страницы установить “Запаролить” (рис. 6).

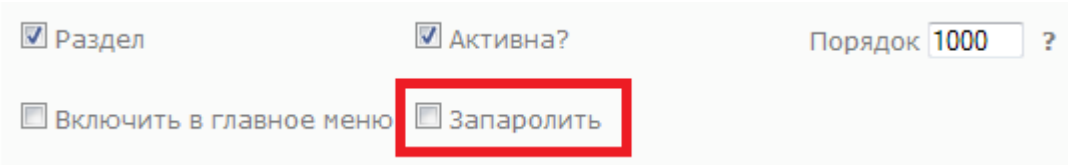

Рис. 6 - Аавторизация на отдельной странице

2.12. Как создать обвязку для новой страницы(контроллер, вьюшка)

1. Для того, чтобы новый контроллер начал обрабатывать запросы, необходимо внести его в дерево маршрутов. Например, если мы хотим, чтобы наш контроллер обрабатывал путь /news/, то мы должны унаследовать наш маршрут от ветки хоста '*' в корне и задать ему паттерн 'news'.
2. Создать новый контроллер в директории /application/controllers/front. Имя класса должно заканчиваться на Controller, например NewsController. Файл должен иметь название <имя класса>.class.php. Класс наследуется от базового класса FrontController, каждый его метод-действие (Index() по умолчанию) должен возвращать объект класса ActionResult.
3. Создать класс для представления (view) в директории /application/views/. Для нашего примера он будет называться NewsView. Для представления действует та же схема именования файлов - NewsView.class.php. Класс

наследуется от базового класса FrontView. Класс представления подключает необходимые шаблоны, передает в шаблонизатор нужные переменные, подключает дополнительные css и js-файлы. Описание процесса создания шаблонов выходит за рамки данного руководства.

4. Опционально. Создать класс-модель. Модели создаются в директории /application/models/ и реализуют методы доступа к данным.

3. Файловая структура

Описание структуры директорий и файлов BitPlatform

index.php

Точка входа приложения. Инициализирует и запускает приложение.

IncPaths.class.php

Класс содержит пути приложения, откуда будут подключаться необходимые модули.

includes.php

Осуществляет подключение необходимых файлов для ускорения работы Autoload.

Директория core

Содержит основные классы, необходимые для работы платформы.

Autoload.class.php

Класс предназначен для автозагрузки файлов приложения через механизм SPL Autoload.

Request.class.php

Класс запроса. Содержит данные GET/POST.

Route.class.php

Класс инкапсулирует маршрут запроса и имя контроллера, обрабатывающего запрос.

Router.class.php

Класс-роутер запросов.

Application.class.php

Класс реализует основные функции приложения - инициализацию, получение маршрута, создание объекта-контроллера, выполнение результата.

Benchmark.class.php

Вспомогательный класс для измерения памяти и времени выполнения участков кода.

Controller.class.php

Абстрактный класс контроллера.

EventDispatcher.class.php

Диспетчер событий BitPlatform.

ObjectsCache.class.php

Класс предназначен для сериализации данных с последующим сохранением в файл.

PluginsSubscriber.class.php

Класс предназначен для подписки плагинов на события

RoutingTableProvider.class.php

Предоставляет таблицу маршрутизации.

Singleton.abstract.php

Абстрактный класс, наследуемый синглтонами.

ViewData.class.php

Класс используется для хранения данных представлений.

ActionResult.abstract

Абстрактный класс результата выполнения контроллера.

Директория interfaces

Содержит интерфейсы, используемые в ядре платформы.

IApplication.interface.php

Интерфейс класса-приложения.

IRouter.interface.php

Интерфейс роутера запросов.

IEventDisatcher.interface.php

Интерфейс диспетчера событий.

Директория exceptions

Содержит файлы классов-исключений, используемых в BitPlatform.

Директория actionResults

Содержит классы результатов выполнения контроллеров (Action Result), наследуемые от абстрактного класса ActionResult.

Директория application

Содержит настройки приложения, контроллеры, модели и представления.

AppSettings.class.php

Содержит настройки приложения: пути до директорий с css, js, изображениями, шаблонами и т.д.

CMSApplication.class.php

Класс, реализующий приложение. Метод Init() так же содержит настройки подключения к БД, логирования и т.п.

EventListenersList.class.php

Класс содержит список обработчиков, подписываемых на события платформы.

RoutingTable.class.php

В файле хранится таблица маршрутизации сайта.

TableNames.class.php

Содержит названия служебных таблиц БД платформы.

Директория controllers

Содержит контроллеры приложения. Для логического разделения контроллеры помещаются в поддиректории admin и front, но это не является обязательным.

Директория eventListeners

Содержит обработчики событий платформы.

Директория models

Содержит классы моделей приложения.

Директория views

Содержит классы-представления приложения. Представления фронт-енда находятся непосредственно в директории, а для административной части - в поддиректории admin.

Директория cache

Содержит кэш-файла автозагрузчика классов.

Директория compile

Содержит кэш-файлы настроек сайта.

Директория templates_c

Содержит скомпилированные шаблоны Smarty.

Директория libs

Содержит библиотеки, необходимые для работы самой платформы, а так же сторонние библиотеки.

Директория Authorisation

Содержит библиотеки авторизации и работы с паролями.

Authorisation.class.php

Класс отвечает за авторизацию пользователя методами POST и Cookie.

Password.class.php

Класс применяется для создания/изменения пароля пользователя и проверки токена смены пароля.

PasswordHash.class.php

Библиотека для работы с хэшами паролей.

Директория BitPlatformMailer

Библиотека (на основе PHPMailer) для создания/отправки сообщений электронной почты.

Директория Cache

Библиотека для кеширования данных.

Директория Captcha

Библиотека для генерации капчи.

Директория CaptchaGoogle

Библиотека для генерации капчи от Google.

Директория ContentManager

Библиотека контент-менеджера. Содержит код генератора форм и элементов управления. Код формы генерируется на основе таблицы БД greeny_fieldsRegistry. Таблица greeny_fieldsRegistry является реестром настроек элементов управления. В ней задается соответствие между полем произвольной таблицы и элементом управления, представляющим это поле. Код генератора выбирает из таблицы настройки и создает элементы управления для создания/редактирования записи для заданной таблицы БД.

Директория Controls

Содержит элементы управления для генератора форм. Каждый элемент управления располагается в своей поддиректории и состоит из файла класса (название класса совпадает с названием элемента управления) и шаблона control.tpl.

ContentManager.class.php

Класс контент-менеджера. Отвечает за создание/изменение/удаление данных таблиц на основе greeny_fieldsRegistry.

Control.class.php

Родительский абстрактный класс для элементов управления. Любой элемент управления должен наследоваться от этого класса. Класс содержит два абстрактных метода - InitControl() выполняется при создании формы и PostBackHandler() - при получении формы методом POST.

ControlFactory.class.php

Класс-фабрика для создания элементов управления на основе параметров из реестра элементов управления.

DBControlsRegistryProvider.class.php

Провайдер настроек элементов управления. Получает настройки по названию формы(таблицы).

IControlsRegistryProvider.class.php

Интерфейс провайдера настроек элементов управления.

FormGenerator.class.php

Класс генерирует массив элементов управления для последующей обработки в шаблоне.

IFlushingControl.interface.php

Интерфейс для элементов управления, которые записывают себя сами. Применяет для сложных полей (когда запись может вестись в несколько таблиц) и абстрактных полей.

includes.php

Файл подключает необходимые классы для работы контент-менеджера.

Директория DBManager

Содержит классы для доступа к БД.

DB.class.php

Класс инкапсулирует подключение к БД через библиотеку PDO.

DBTableManager.class.php

Инкапсулирует работу с таблицей БД, реализует простые операции - select, insert, update, delete.

DBTableStructure.class.php

Получает и хранит информацию о структуре полей таблицы БД.

DBFieldStructure.class.php

Хранит информацию о поле таблицы - тип, размер и т.д.

Директория Debug

Содержит классы отладки.

Log.class.php

Класс пишет лог в файл (по умолчанию php.log).

DBLog.class.php

Пишет лог в специальную таблицу БД (по умолчанию 'greeny_log').

Директория FileManager

Классы для работы с файловой системой.

FileManager.class.php

Реализует работу с файлами - получение информации, удаление, перемещение и т.п.

ImageManager.class.php

Реализует работу с изображениями - загрузку на сервер, изменение размеров, копирование, удаление.

Директория Modules

Modules.class.php

Класс-менеджер модулей.

Директория PHPMailer

Содержит библиотеку PHPMailer для работы с электронной почтой.

Директория Pages

Содержит классы для работы со страницами сайта.

Page.class.php

Класс страницы. Реализует получение страницы из БД, получение алиаса страницы, мета-информации и контента.

PageStructure.class.php

Класс реализует управление деревом страниц.

PagesTypes.class.php

Получает из БД константы типов страниц.

Директория RSS

RSSGenerator.class.php

Класс используется для генерации RSS 2.0.

Директория SettingsManager

Менеджер настроек сайта.

DBSettingsManager.class.php

Менеджер настроек сайта из БД.

CachingDBSettingsManager.class.php

Расширяет класс DBSettingsManager, кеширует настройки в файл.

DBSettingsProvider.class.php

Провайдер настроек из БД.

ISettingsProvider.class.php

Интерфейс провайдера настроек.

Директория Smarty

Библиотека шаблонизатора Smarty.

Директория Structure

Библиотека управления древовидной структурой сайта.

MainMenuManager.class.php

Менеджер главного меню сайта. Строит дерево, добавляет/удаляет страницы в дереве меню.

SiteStructure.class.php

Менеджер древовидной структуры страниц сайта. Позволяет удалять, добавлять, редактировать, получать алиасы страниц из структуры сайта.

Директория modules

Содержит подключаемые модули платформы. Каждый модуль располагается в своей поддиректории, имя которой совпадает с названием модуля. В директории модуля располагаются следующие файлы:

- <имя модуля>.class.php - собственно класс модуля;

- <имя модуля>.admin.php - класс, описывающий страницу админки;

- install.xml - XML-файл установки модуля;

- другие файлы, необходимые для функционирования модуля.

Формат XML-файла модуля

Основная ветка <module> содержит обязательные узлы <name> и <description>, значения которых представляют название и текстовое описание модуля соответственно.

Ветка <dbtables> содержит узлы <dbtable>, в каждом из которых содержится SQL-код создания таблиц. Данные таблиц могут располагаться в ветке <dbdata>, формат которой следующий: имя дочерней ветки совпадает с названием заполняемой таблицы (по одной ветке на запись, т.е. для добавления 3х записей в таблицу необходимо указать соответствующую ветку 3 раза). В ветке с именем таблицы располагаются узлы, названия которых определяют имена заполняемых полей, а значения - соответственно значения заполняемых полей.

Опциональная ветка <pagetypes> содержит подветки <pagetype>, в которых располагается описания создаваемых типов страниц для модуля. Имена узлов совпадают с именами полей таблицы 'greeny_pageTypes' - name, description, handlerType и constantName.

Настройки, необходимые для работы модуля могут содержаться в ветке <settings>, каждая подветка которой, <setting> содержит данные о настройке. <setting> имеет следующие узлы:

- <name> - имя настройки;

- <value> - значение настройки;

- <description> - описание для пользователя;

- <order> - порядок в списке настроек.

CommonModule.class.php

Общий базовый класс для всех модулей.

Директория publicLibs

Содержит клиентские js-библиотеки, используемые фронт-ендом и бэк-ендом (jQuery, FCKEditor и т.п.).

Директория scripts

Содержит js-скрипты для работы приложения.

Директория sections

Содержит секции старой версии BitPlatform. В новой версии секции были заменены на страницы, но административная часть в нескольких местах содержит вызовы секций, которые необходимо удалить в будущем.

Директория templates

Содержит шаблоны Smarty для платформы.

Директория admin

Шаблоны для административной части сайта.

Директория front

Шаблоны для фронт-части сайта. Шаблоны могут группироваться в директории для удобства.

Директория uploaded

Предназначена для загрузки файлов/изображений пользователями.

4. Установка на хостинге

Для установки BitPlatform на хостинге необходимо:

1. Обеспечить доступ к созданной на хостинге Базе Данных (например, с использованием phpmyadmin или любого другого менеджера управления базами данных) или «ssh» доступ к серверу.
2. Доступ посредством протоколов «ftp» или «ssh», или любой другой файловый менеджер для доступа к файлам на хостинге.
3. В файле проекта /application/config/config.php необходимо прописать доступы к Базе Данных в соответствующих переменных

   DB::$DBName = 'db_name'; // Имя БД

   DB::$DBUserName = 'db_user'; // имя пользователя БД

   DB::$DBPassword = 'user_password'; // пароль

   DB::$DBHost = 'localhost'; // хост сервера базы данных (если он отличается от localhost)

4. В системе управления Базой Данных необходимо подключиться к базе и

импортировать в нее SQL файл, находящийся в корневой папке файловойго архива БитПлатформы «/install»

5. С помощью файлового менеджера (см. п.2) переместить файлы проекта в корень (начальную, корневую папку) сайта на хостинге.
6. С помощью встроенных приложений установить рекурсивно доступ на запись папкам «/compile» и «/uploaded».