3 Реализация

3.1 Логическая структура web-приложения

Предметной областью данного дипломного проекта является деятельность свадебного салона «Венера».

Разрабатываемое в рамках дипломного проекта программное средство должно выполнять следующие функции:

* хранение и отображение информации о деятельности салона;
* возможность просмотра имеющихся в наличии свадебных платьев, украшений для автомобилей и аксессуаров;
* возможность поиска информации по предоставляемой продукции;
* возможность добавления, удаления и редактирования информации в режиме «администратор», содержащейся на сайте;
* возможность организации обратной связи.

На основании всего вышеперечисленного формируется модель предметной области. Потенциальные клиенты, использующие данное программное средство будут иметь возможность получать всю необходимую информацию, касающуюся деятельности свадебного салона «Венера».

Достоинством организации данной структуры проекта является необязательность выполнения каждого из шагов, возможность возвращаться к пройденным ранее блокам и пропускать некоторые информационные разделы.

Логическая структура сайта представляет собой набор тематических разделов, связанных между собой и включает в себя шесть основных тематических страниц, дающих возможность удобного использования и быстрого доступа к необходимой информации:

* «Главная»;
* «Свадебные платья»;
* «Украшения авто»;
* «Аксессуары»;
* «Полезные советы»;
* «Контакт».

Каждая из страниц содержит ссылки на другие разделы сайта.

3.2 Физическая структура web-приложения

Для организации физической модели данных будут созданы следующие таблицы:

Таблица 1 - Администратор

|  |  |  |
| --- | --- | --- |
| Ключ | Имя | Тип |
| \* | id | int(11) |
|  | login | varchar(255) |
|  | password | varchar(255) |

В таблице «Администратор» будет содержаться вся информация об администраторе: логин и пароль для входа в администраторскую часть сайта.

Таблица 2 - Советы

|  |  |  |
| --- | --- | --- |
| Ключ | Имя | Тип |
| \* | id | int(11) |
|  | name | varchar(255) |
|  | text | text |

В данной таблице будут содержаться все добавленные на сайт полезные при подготовке к свадьбе советы.

Таблица 3 – Категория

|  |  |  |
| --- | --- | --- |
| Ключ | Имя | Тип |
| \* | id | int(11) |
|  | name | varchar(255) |
|  | text | varchar(255) |

В таблице «Категория» содержится перечень категорий товаров таких, как: платья, украшения авто, аксессуары.

Таблица 4 – Товары

|  |  |  |
| --- | --- | --- |
| Ключ | Имя | Тип |
| \* | id | int(11) |
|  | name | varchar(255) |
|  | photo | varchar(255) |
|  | category\_id | int(11) |

В таблице «Товары» содержится перечень всех товаров, представленных в салоне на продажу или прокат.

В данном дипломном проекте используется популярная в настоящее время и диктуемая Yii архитектура MVC. MVC направлен на отделение бизнес-логики от пользовательского интерфейса, чтобы разработчики могли легко изменять отдельные части приложения не затрагивая другие. В архитектуре MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс (например, текст, поля ввода), а контроллер обеспечивает взаимодействие между моделью и представлением.

Входной скрипт – это PHP-скрипт, выполняющий первоначальную обработку пользовательского запроса. Это единственный PHP-скрипт, который доступен для исполнения конечному пользователю по прямому запросу.

Сначала скрипт подключает файл инициализации фреймворка yii.php, затем создает экземпляр приложения с установленными параметрами и запускает его на исполнение.

Приложение может выполняться в отладочном (debug) или рабочем (production) режиме в зависимости от значения константы YII\_DEBUG.

По умолчанию её значение установлено в false, что означает рабочий режим. Для запуска в режиме отладки установите значение константы в true до подключения файла yii.php. Работа приложения в режиме отладки не столь эффективна из-за ведения множества внутренних логов. С другой стороны, данный режим очень полезен на стадии разработки, т.к. предоставляет большее количество отладочной информации при возникновении ошибок.

Приложение (application) – это контекст выполнения запроса. Основная задача приложения – предварительная обработка запроса (request resolving) пользователя и передача его соответствующему контроллеру для дальнейшей обработки. Другой задачей приложения является хранение параметров конфигурации уровня приложения (application-level configuration). Поэтому приложение также называютфронт-контроллером.

Приложение создается входным скриптом как одиночка (singleton). Экземпляр приложения доступен из любой его точки посредством Yii::app().

По умолчанию, приложение – это экземпляр класса CWebApplication, который может быть настроен с использованием конфигурационного файла (или массива). Необходимые значения свойств устанавливаются в момент создания экземпляра приложения. Альтернативный путь настройки приложения – расширение класса CWebApplication.

Конфигурация – это массив пар ключ-значение, где каждый ключ представляет собой имя свойства экземпляра приложения, а значение - начальное значение соответствующего свойства. Например, следующая конфигурация устанавливает значения свойств приложения name и defaultController:

Функциональность приложения может быть легко модифицирована и расширена благодаря компонентной архитектуре. Приложение управляет набором компонентов, каждый из которых имеет специфические возможности. Например, приложение производит предварительную обработку запроса пользователя, используя компоненты CUrlManager и ChttpRequest.

Далее перечислены ключевые компоненты, предопределенные классом CwebApplication:

* assetManager: CAssetManager – управляет публикацией файлов ресурсов (asset files);
* authManager: CAuthManager – контролирует доступ на основе ролей (RBAC);
* cache: CCache – предоставляет возможности кэширования данных; следует учесть что следует указать используемый класс (например, CMemCache, CDbCache), иначе при обращении к компоненту будет возвращен null;
* clientScript: CClientScript – управляет клиентскими скриптами (javascripts и CSS);
* coreMessages: CPhpMessageSource – предоставляет переводы системных сообщений Yii-фреймворка;
* db: CDbConnection – обслуживает соединение с базой данных; обратите внимание, что для использования компонента необходимо установить свойство connectionString;
* errorHandler: CErrorHandler – обрабатывает не пойманные ошибки и исключения PHP;
* format: CFormatter – форматирует данные для их последующего отображения.
* messages: CPhpMessageSource – предоставляет переводы сообщений, используемых в Yii-приложении;
* request: CHttpRequest – предоставляет информацию, относящуюся к пользовательскому запросу;
* securityManager: CSecurityManager – предоставляет функции, связанные с безопасностью (например, хеширование, шифрование);
* session: CHttpSession – обеспечивает функциональность, связанную с сессиями;
* statePersister: CStatePersister – предоставляет метод для сохранения глобального состояния;
* urlManager: CUrlManager – предоставляет функции парсинга и формирования URL;
* user: CWebUser – предоставляет идентификационную информацию текущего пользователя;
* themeManager: CThemeManager – управляет темами оформления.

Жизненный цикл приложения при обработке пользовательского запроса:

1. Предварительная инициализация приложения через CApplication::preinit().
2. Инициализация автозагрузчика классов и обработчика ошибок.
3. Регистрация компонентов ядра.
4. Загрузка конфигурации приложения.
5. Инициализация приложения CApplication::init():

1) регистрация поведений приложения;

2) загрузка статических компонентов приложения;

1. Вызов события onBeginRequest.
2. Обработка запроса:

1) разбор запроса;

2) создание контроллера;

3) запуск контроллера на исполнение;

1. Вызов события onEndRequest.

Контроллер (controller) – это экземпляр класса CController или производного от него. Контроллер создается приложением в случае, когда пользователь его запрашивает. При запуске контроллер выполняет соответствующее действие, что обычно подразумевает создание соответствующих моделей и рендеринг необходимых представлений. В самом простом случае действие – это метод класса контроллера, название которого начинается на action.

Модель (model) – это экземпляр класса CModel или производного от него. Модель используется для хранения данных и применимых к ним бизнес-правил.

Модель представляет собой отдельный объект данных. Это может быть запись таблицы базы данных или форма пользовательского ввода. Каждое поле объекта данных представляется атрибутом модели. Каждый атрибут имеет метку и может быть проверен на корректность, используя набор правил.

Yii предоставляет два типа моделей: модель формы (form model) и Active Record. Оба типа являются расширением базового класса CModel.

Модель формы – это экземпляр класса CFormModel. Она используется для хранения данных, введенных пользователем. Как правило, мы получаем эти данные, обрабатываем, а затем избавляемся от них. Например, на странице авторизации модель такого типа может быть использована для представления информации об имени пользователя и пароле. Подробное описание работы с формами приведено в разделе Создание формы.

Active Record (AR) – это шаблон проектирования, используемый для абстрагирования доступа к базе данных в объектно-ориентированной форме. Каждый объект AR является экземпляром класса CActiveRecord или производного от него, представляя отдельную запись в таблице базы данных. Поля строки представлены свойствами AR-объекта.

Представление – это PHP-скрипт, состоящий преимущественно из элементов пользовательского интерфейса. Он может включать выражения PHP, однако рекомендуется, чтобы эти выражения не изменяли данные и оставались относительно простыми. Следуя концепции разделения логики и представления, большая часть кода логики должна быть помещена в контроллер или модель, а не в скрипт представления. У представления есть имя, которое используется, чтобы идентифицировать файл скрипта представления.

Имя представления должно совпадать с названием файла представления. К примеру, для представления edit соответствующий файл скрипта должен называться edit.php. Чтобы отрендерить представления необходимо вызвать метод CController::render(), указав имя представления. При этом метод попытается обнаружить соответствующий файл представления в директории protected/views/ControllerID.

Yii-приложения состоят из компонентов–объектов, созданных согласно спецификациям. Компонент (component) – это экземпляр класса CComponent или производного от него. Использование компонента в основном включает доступ к его свойствам, а также вызов и обработку его событий. Базовый класс CComponent устанавливает то, как определяются свойства и события.

Свойство компонента схоже с публичной переменной-членом класса (public member variable). Мы можем читать или устанавливать его значение.

Модуль – это самодостаточная программная единица, состоящая из моделей, представлений,контроллеров и иных компонентов. Во многом модуль схож с приложением. Основное различие заключается в том, что модуль не может использоваться сам по себе – только в составе приложения. Пользователи могут обращаться к контроллерам внутри модуля абсолютно так же, как и в случае работы с обычными контроллерами приложения.

Модули могут быть полезными в нескольких ситуациях. Если приложение очень объемное, мы можем разделить его на несколько модулей, разрабатываемых и поддерживаемых по отдельности. Кроме того, некоторый часто используемый функционал, например, управление пользователями, комментариями, может разрабатываться как модули, чтобы впоследствии можно было с легкостью воспользоваться им вновь.

3.3 Описание структуры и формата страниц

Если открыть любую веб-страницу, то она будет содержать в себе типичные элементы, которые не меняются от вида и направленности страницы:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Пример</title>

</head>

<body>

</body>

</html>

Кроме того все страницы в приложении разделены на группы, которые имеют общий шаблон, меняется только внутренняя часть страницы. Так были разработаны следующие слои:

* слой для контекстных страниц клиентской части;
* слой для администраторской части.

В администраторской части для большей интерактивности используются диалоговые окна, создаваемые с помощью библиотеки jQuery UI.

Данное программное средство содержит 3 основные папки: js, css, img.

В папке js находятся страницы jquery.js, используемые для взаимодействия HTML и JavaScript .

В папке css лежат CSS-стили необходимые для корректной работы верстки страниц.

В папке img расположены графические элементы, используемые в разрабатываемом web приложении.

Каждая страница web-приложения снабжена элементами навигации (ссылками) для удобства перемещения и может работать независимо от других страниц.

Работа с программой начинается с загрузки главной страницы index.php, откуда пользователь может получить доступ к другим страницам интернет-сайта посредством ссылок, организующих переход на эти страницы.

При нажатии на ссылку «Свадебные платья» открывается страница с каталогом свадебных платьев, доступных для продажи или проката.

<div class="menu\_item">

<a href=dress.php"/">Свадебные платья </a>

</div>

При нажатии на ссылку «Украшения авто» осуществляется переход на страницу с фотографиями украшений для автомобилей, доступных для продажи или проката.

<div class="menu\_item">

<a href="auto.php">Украшения авто</a>

</div>

При нажатии на ссылку «Аксессуары» осуществляется переход на страницу с каталогом аксессуаров к свадебному платью.

<div class="menu\_item">

<a href=" accessories.php"> Аксессуары </a>

</div>

При нажатии на ссылку «Полезные советы» осуществляется переход к информации о подготовке торжества.

<div class="menu\_item">

<a href="tips.php">Полезные советы </a>

</div>

В верхней части каждой страницы есть меню с ссылками на каждую из других страниц.

3.4 Описание используемых библиотек и элементов управления

В данном приложении очень тесно реализована интеграция с библиотекой jQuery и производной от нее библиотекой jQuery UI.

jQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API по работе с Ajax.

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки, управление передаётся JQuery, идентифицирующей кнопки и затем преобразовывающий его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функциональность, полезную для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, бо́льшая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно ту JavaScript-функциональность, которая на нём была бы востребована.