Урок 8

Практика. Реализация движка интернет-магазина

На этом уроке мы подведём итог пройденному материалу, завершив работу над движком, функционал которого мы наращивали в течение курса

Архитектура

Наше приложение уже имеет базовый функционал, способный авторизовывать пользователя, давать ему возможность просматривать страницы сайта и оставлять отзывы. Но это ещё не всё. Чего же нам не хватает?

1. Возможности оформления заказа (иначе какой же это интернет-магазин?).
2. Автозагрузки файлов библиотек (согласитесь, неудобно каждый раз подключать новые или писать всё в одном файле).
3. Асинхронности (AJAX делает процесс работы с сайтом гораздо приятнее).
4. Структуры шаблонов.

Итак, обо всём по порядку.

Оформление заказа

На данный момент у нас есть корзина, в которую можно положить товары. Записей в корзине много и надо как-то скомпоновать их в заказ. Поэтому мы вводим ещё один уровень, находящийся выше корзины, и это заказ.
Сущность заказа будет содержать в себе его общую сумму, id оформившего данный заказ клиента и техническую информацию. После того как мы получим id заказа, нужно занести его во все соответствующие корзине пользователя записи, параллельно пометив их как вошедшие в заказ (is_in_order). Согласитесь, у одного покупателя на один момент времени может быть только одна корзина.

Таким образом, алгоритм создания заказа таков:

1. Сбор информации о корзине пользователя.
2. Подсчёт суммы заказа.
3. Внесение данных в БД с получением ID заказа.
4. Занесение ID заказа в корзину.
5. Пометка товаров в корзине как купленных.
6. Завершение процесса.

Помимо этого нужно учесть ещё две важные вещи:

1. Пользователь должен уметь видеть свои заказы в личном кабинете и иметь ограниченный функционал управления ими (отмена, например).
2. Администратор должен уметь видеть все заказы в обратном хронологическом порядке.

Автозагрузка файлов библиотек

До текущего момента нам приходилось делать две неудобные вещи:

1. Писать много кода в одном и том же файле.
2. Каждый раз использовать require при разделении кода на файлы.

Это очень неудобно, поскольку при работе нескольких разработчиков в один прекрасный момент кто-нибудь может ошибиться или что-то забыть – по этой причине функционал перестанет нормально работать.

Мы исправим это. Что нужно сделать:

1. Согласно архитектурному концепту все файлы подключаемых библиотек лежат в папке engine.
2. Нужно обойти её при помощи scandir().
3. Все файлы, имеющие расширение .lib.php, будут автоматически включены в код проекта.

Теперь можно распределять логику по соответствующим файлам, не смешивая, например, функции оформления заказов с аутентификацией пользователей.

Асинхронность

Про плюсы асинхронности вы могли слышать на курсе JavaScript. Она позволяет не только расширить функциональность отдельно взятой страницы, улучшив её юзабилити, но и создавать мощные и удобные одностраничные приложения.

Мы будем использовать библиотеку jQuery, упрощающую работу с AJAX.

```
$(селектор).on(событие, function(){
      $.ajax({
         url: адрес серверного обработчика,
         type: тип HTTP запроса,
         data: пакет передаваемых данных (для POST)
         error: обработчик серверных ошибок,
         success: обработчик успешного (с точки зрения сервера) завершения,
         dataType : тип данных в ответе
      })
  });
```

Структура шаблонов

На сайте есть много общих элементов. Как минимум, это общий скелет, подвал и шапка. Общую структуру мы вынесем в файл skeleton.tpl.

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8" />
   <title>{{TITLE}}</title>
   <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
   <script src="/js/engine.js" type="text/javascript"></script>
</head>
<body>
<div id="wrapper">
   <div id="header">
      {{HEADER}}
   </div>
   <div id="content">
      {{TPLCONTENT}}
   </div>
   <div id="footer">
```

```
        {{FOOTER}}
    </div>
</div>
</body>
</html>
```

В переменные HEADER, TPLCONTENT и FOOTER будут приходить соответствующие подшаблоны. Этот момент нужно поддержать и на уровне логики.
Подобный подход позволяет очень удобно работать с SEO и иметь гарантированно общий стиль сайта.

Итоги

По итогам курса мы получили полнофункциональный сайт интернет-магазина, который уже можно применять для работы с пользователями.

Практическое задание

1. Создать функционал асинхронного контроля заказов администратором.
2. Создать функционал управления товарами.
3. Создать функционал асинхронного контроля заказов пользователем.
4. *Изменить хранение цен в системе таким образом, чтобы они могли меняться в зависимости от времени и действующих акций.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Изучаем PHP и MySQL – Линн Бейли, Майкл Моррисон.
2. “PHP 5 в подлиннике” – Д.Котеров.