Введение

Интернет покупки и онлайн магазины появились сравнительно недавно, но, несмотря на это уже прочно вписались в нашу жизнь.

Это удобно, так как нет необходимости физически обходить магазины и вручную искать товар - это можно сделать онлайн в один клик. Однако, возникает другая проблема - необходимость поиска и посещения интернет-магазинов, что при некоторых обстоятельствах может занять больше времени, чем посещение офлайн магазинов.

В связи с этим возникает необходимость в агрегации и составления некоторого каталога товаров, информация в который будет поступать из определённого количества магазинов с возможностью добавления новых.

Агрегация подразумевает обязательное получение информации с интернет-магазина источника. К сожалению, далеко не все сайты могут предоставить интерфейс доступа к базе данным, поэтому необходимо использовать другие методы, например, синтаксический анализ.

1 Теоретическая часть

1.1 Синтаксический анализ как инструмент для получения информации

Синтаксический анализ сайта - это синтаксический анализ информации на интернет странице, которая состоит из иерархического набора данных [8]. В результате необходимо извлечь необходимую нам информацию. Выполняется специальной программой – парсером.

Синтаксический анализ используется в следующих случаях:

- отсутствие прямого доступа к базе данных сайта;

- большие объемы данных;

- частое обновление данных.

Анализ информации на сайте возможен и вручную при небольшом требовании к количеству информации. Однако программа – парсер имеет следующие преимущества над ручным анализом:

- скорость. Программе необходимо несколько миллисекунд (не считая времени, необходимого для загрузки странице из интернета) для обработки информации, а человеку как минимум 5 – 10 секунд;

- коэффициент ошибок. В программе он стремится к нулю при отсутствии ошибок в коде страницы, у человека же он всегда присутствует;

- эффективность. Программа может не только получить требуемые данные, но и представить в нужном виде. Это может текстовый документ, таблица, база данных. В ручном режиме это займет примерно минимум столько же времени, сколько и на поиск и анализ.

Программа – анализатор выполняет следующие этапы [8]:

- получение исходного кода страницы. Выполнятся различными способами: как использование готовых функций (например file_get_contents() в PHP), так и написание собственного веб – клиента;

- извлечение из html – кода необходимых данных. После получения страницы необходимо обработать её – отделить текстовые данные от

гипертекстовой разметки, построить иерархическое дерево документа (Document Object Model), извлечь нужную информацию;

- сохранить результат. После обработки остается сохранить данные в виде файла, иерархического дерева, базы данных.

Сложность парсинга сайта заключается еще в том, что необходимо анализировать сразу несколько страниц сайта, а зачастую и все. В этом случае есть следующие методы для обхода нужных страниц сайта [10]:

1) Необходимо находить и извлекать не только нужную нам информацию, но и гиперссылки на внутренние страницы.
2) Проследить логику формирования ссылки для страниц. Это может быть целочисленное значение, либо имена страниц. Далее, по полученной логике сформировать адреса.
3) Ручной обход. Все адреса берутся из всех посещенных страниц и передаются в программу.

1.2 Выбор библиотеки для синтаксического анализа

Для того, чтобы приступить к синтаксическому анализу, необходимо выбрать библиотеку.

Главный критерий выбора – совместимость с веб-приложениями, а это значит, что библиотека должна быть написана на языке PHP для легкого встраивания. Второй немаловажный критерий – понятный синтаксис и документация.

По данным двум критериям я решил использовать библиотеку «PHP Simple HTML DOM Parser».

Эта библиотека поставляется в виде отдельного PHP файла, а это значит, что ее можно использовать для любого веб-приложения на языке PHP. Также, она имеет понятный синтаксис. Для загрузки страницы используется стандартная функция file_get_html(). Для поиска нужной нам информации

используется функция find(), в которую можно передать id блока, класс, атрибут.

1.3 Контроллеры

Контроллер – это функция, служащая для получения информации из HTTP запроса и создающая на ее основе HTTP ответ в виде объекта Response. Контроллер содержит логику приложения, необходимую для отображения страницы.

Жизненный цикл контроллера состоит из следующих этапов:

1) Каждый запрос обрабатывается одним фронт – контроллером, который загружает приложение.

2) Router читает информацию из запроса, ищет подходящий маршрут и получает параметр _controller из маршрута.

3) Контроллер, соответствующий маршруту, выполняется и его код формирует объект Response.

4) HTTP – заголовки и контент объекта Response отправляются обратно к клиенту, отправившему изначальный запрос.

1.4 Маршрутизация

Маршрутизатор позволяет определить URL, который мы можем привязать к различным частям приложения. Symfony поддерживает следующие форматы задания маршрутов:

- YAML;
- PHP;
- XML.

Цель системы маршрутизации – анализ URL и определение того, какой контроллер должен быть выполнен. Процесс выглядит так:

1) Запрос обрабатывается фронт контроллером Symfony.

2) Ядро Symfony вызывает маршрутизатор для анализа запроса.

3) Маршрутизатор устанавливают соответствие между входящим маршрутом и возвращает информацию о маршруте, включая данные о том, какой контроллер требуется выполнить.

4) Ядро Symfony выполняет контроллер, который в конечном итоге возвращает объект Response.

Процесс маршрутизации показан на рисунке 1 [4].

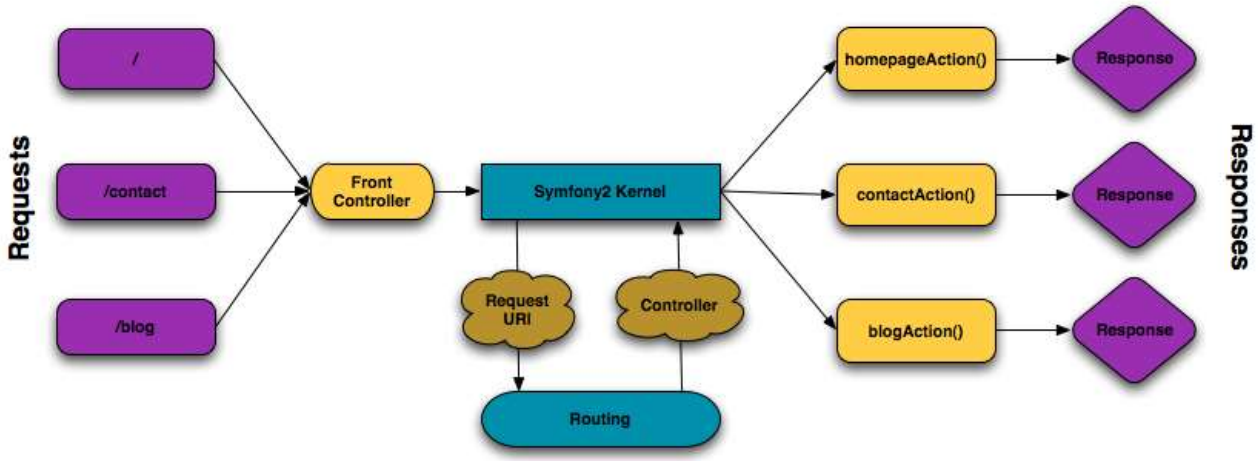

Рисунок 1 – процесс маршрутизации

1.5 Шаблоны

Шаблон – инструмент для отделения визуального представления и содержимого. В состав Symfony входит мощный язык шаблонов Twig, позволяющий создавать лаконичные шаблоны, которые будут удобнее и функциональнее PHP шаблонов [13].

Система шаблонов Twig задумана для быстрого создания представлений. Преимущества Twig перед PHP:

- полноценное наследование шаблонов;
- контроль пробельных символов;
- контроль выполнения подозрительного кода;
- расширение пользовательскими фильтрами и функциями;

- быстрота, обеспечиваемая за счет кэширования.

Шаблоны могут располагаться в двух различных местах:

- app/Resources/views/: Директория views может содержать шаблоны, общие для всего приложения (например layout приложения), а также шаблоны, которые переопределяют шаблоны пакетов (см. Переопределение шаблонов пакета);

- путь к пакету/ Resources/ views/: Каждый пакет содержит свои собственные шаблоны в директории Resources/views (и её поддиректориях). Большинство шаблонов будет располагаться внутри пакета.

1.6 Базы данных и Doctrine

Библиотека Doctrine позволяет работать с базой данных не с точки зрения строк и столбцов, а с точки зрения объектов. Для того чтобы создать эти объекты, необходимо создать класс. В этот класс необходимо поместить переменные, которые впоследствии будут переданы в базу данных.

Однако такого набора действий недостаточно для работы с базой данных. Библиотеке Doctrine необходимы метаданные, сообщающие ей то, как соответствующие поля класса будут отображены в базу данных. Их можно указать в классе через аннотации.

После добавления метаданных необходимо создать методы доступа для полей класса. Это можно сделать как вручную, так и выполнив следующую команду: « php app/console doctrine:generate:entities Acme /bundlename /Entity /Name».

Наконец, остается создать таблицу и схему базы данных. Данное действие можно осуществить при помощи команды «doctrine:schema:update – force». Эта команда сравнивает как должна выглядеть база данных (основываясь на информации об отображении для сущностей) с тем, как она выглядит на самом деле, и создаёт SQL выражения, необходимые для обновления базы данных до того вида, какой она должна быть.

Для сохранения объекта в базу данных нужно создать экземпляр объекта базы данных и при помощи методов доступа задать необходимые значения. Далее, необходимо вызвать метод persist(), который сообщает Doctrine команду на управление объектом, и метод flush(), с помощью которого Doctrine непосредственно сохраняет объект в базу данных. Flush() просчитывает набор изменений и выполняет наиболее эффективный и возможный запрос [11].

Для получения объекта необходимо получить сущности класса при помощи метода getrepository(). Далее, можно воспользоваться одним из разновидности метода find(). Есть возможность сделать выборку всех элементов при помощи метода findall(), либо по определенному полю при помощи метода findbyprice(). Так же можно передать массив в функцию findby() и выборка будет выполнена по каждой паре ключ-значение.

В библиотеке Doctrine есть возможность запросить объект через язык DQL (Data Query Language). Для этого случая существует функция createquery(), в который и передается запрос.

2 Практическая часть

2.1 Постановка задачи

В качестве практической части была поставлена цель создать веб-приложение каталога товаров, где пользователь может выбрать интересующий его продукт, сравнить цены и сделать свой выбор, не выходя из дома.

2.2 Спецификация задачи

Задача – разработать веб-приложение каталога товаров. Информация должна собираться с сайтов интернет магазинов в автоматическом режиме и выводится в удобном виде.

2.3 Структура базы данных

Для корректной работы приложения необходимо обеспечить сохранение, извлечение и обработку данных, поступающих в результате поступления информации о товарах и просмотра пользователями их. Оптимальным решением этой задачи будет использование базы данных. В результате была разработана структура базы данных, показанная на рисунке 2.

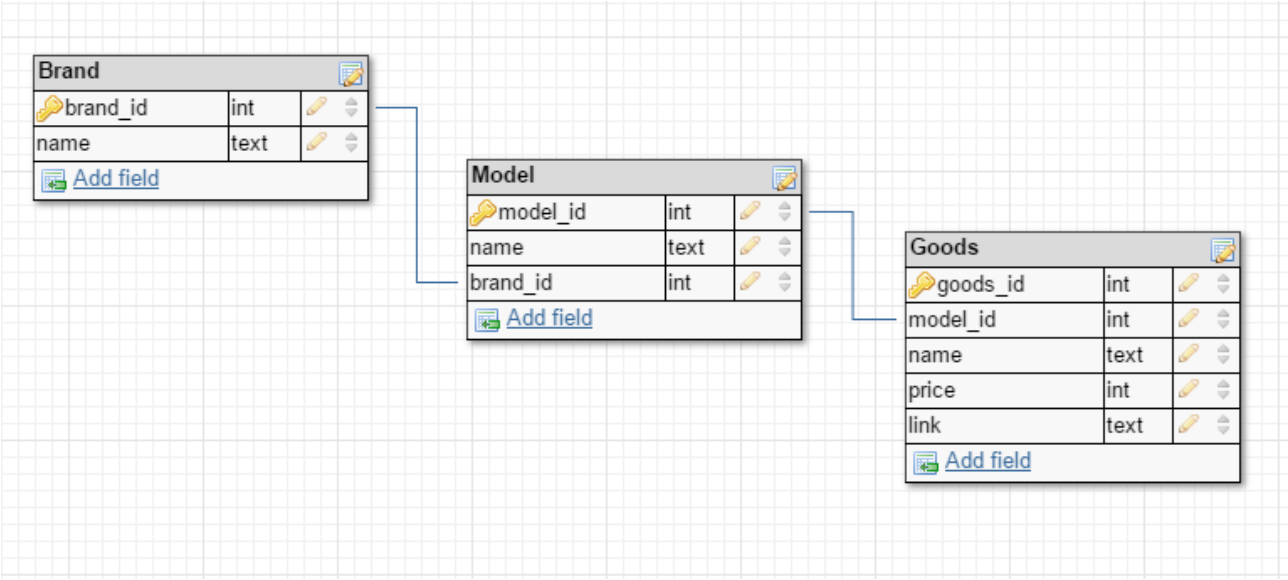

Рисунок 2 – Структура базы данных

Полученная структура состоит из трех таблиц:

1) Brand, содержащая целочисленный идентификационный номер марки (с инкрементным счетчиком), текстовое поле, содержащее название марки.

2) Model, содержащая целочисленный идентификационный номер марки из таблицы Brand, текстовое поле, хранящее имя модели, и целочисленный идентификационный номер модели.

3) Goods, содержащая целочисленный идентификационный номер модели из таблицы Model, текстовое поле, хранящее имя товара, целочисленный идентификационный номер товара, целочисленное значение цены и текстовую гиперссылку, по которой можно приобрести товар.

2.4 Реализация поставленной задачи

2.4.1 Сбор информации

Для источника информации о товарах была выбрана автомобильная тематика и сайт «an-auto.ru», продающий детали для автомобилей в Волгограде.

Была выбрана библиотека «PHP Simple HTML DOM Parser», поставляемая в виде PHP файла.

Для того, чтобы приступить к сбору информации на сайте (парсингу), необходимо проанализировать исходный код и найти блоки и тэги, содержащие нужные нам данные. Так же необходимо выбрать стратегию, по которой будут браться ссылки на остальные страницы и анализироваться.

Главная страница сайта изображена на рисунке 3.

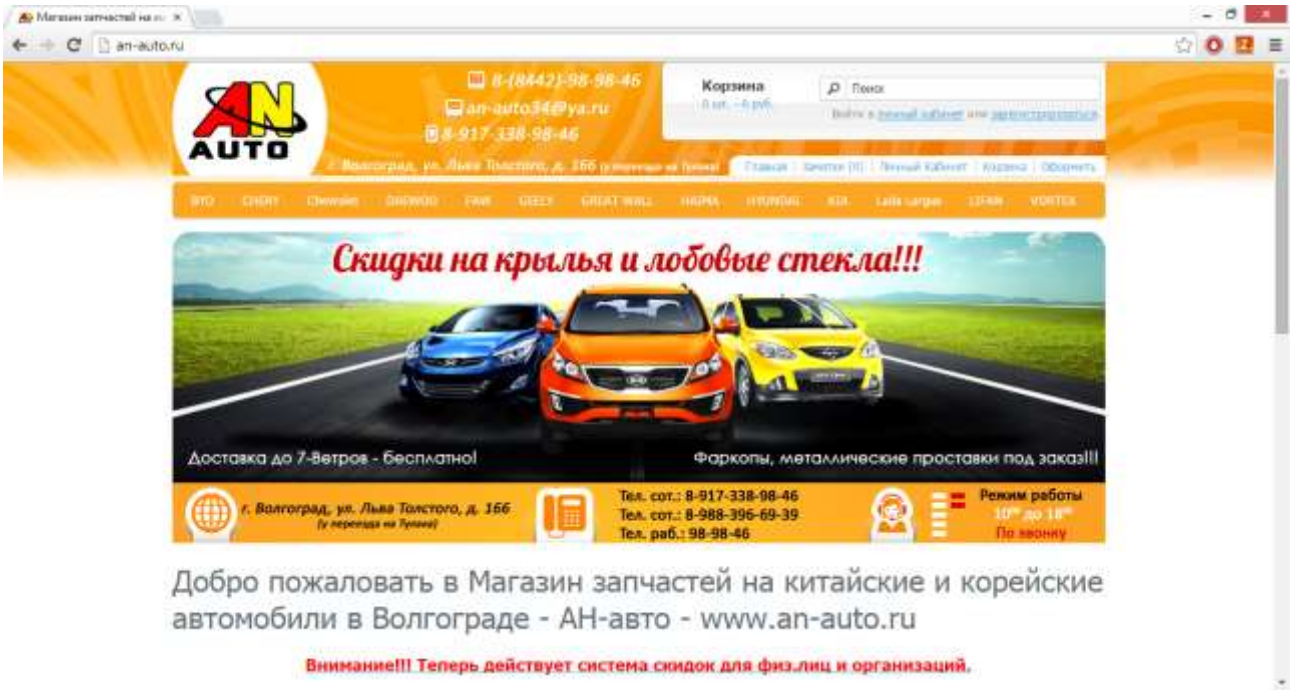

Рисунок 3 – главная страница сайта

Для того, чтобы посмотреть исходный код, необходимо нажать на комбинацию «Ctrl+U», либо в контекстном меню по правому клику мыши выбрать «Просмотр исходного кода страницы» (см. рисунок 4).

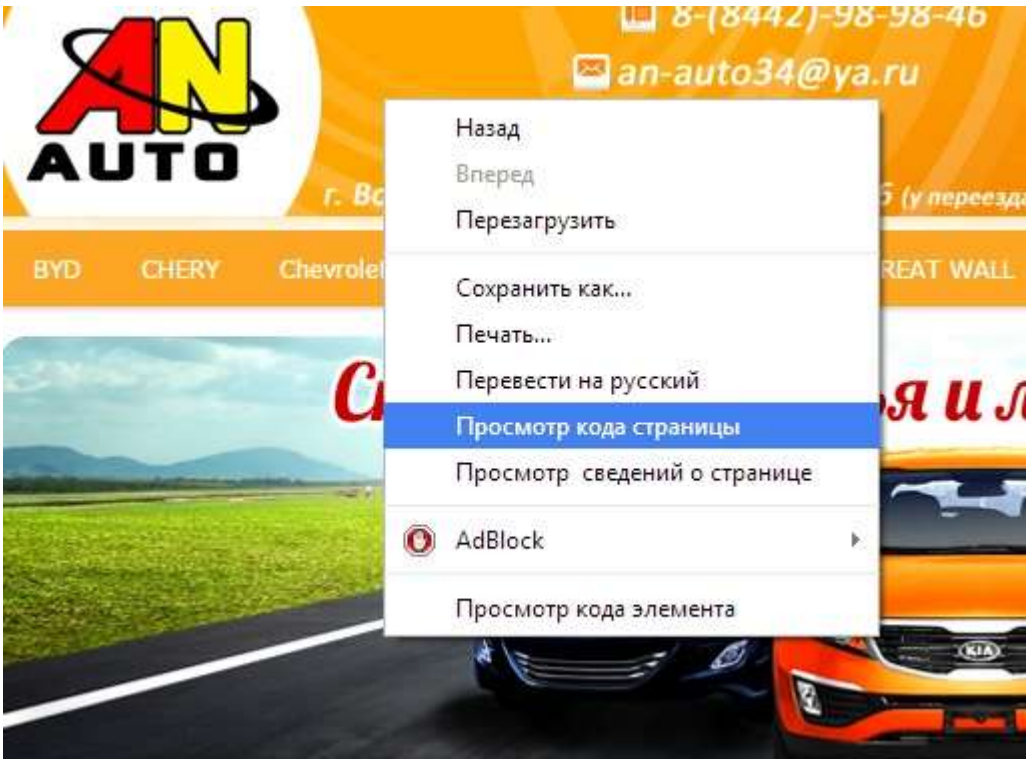

Рисунок 4 – контекстное меню

После одной из этих операций откроется новая вкладка, в которой будет показан исходный код страницы (см. рисунок 5).

Рисунок 5 – исходный код страницы

Далее можно заметить, что в блоке с идентификационным именем «menu» хранятся ссылки на все элементы каталога. Следовательно, мы можем по данному идентификационному номеру получить все эти ссылки и передать на дальнейший анализ каждую. Такой подход избавляет нас от ручного перехода по ссылкам и экономит время.

После получения списка ссылок необходимо проанализировать одну из них, чтобы по аналогии проделать все необходимые операции с остальными.

При открытии одной из ссылок перед нами предстает список товаров. Сразу ставится задача – не только извлечь весь этот список, но и соотнести его с соответствующей категорией – маркой и моделью.

После просмотра исходного кода можно заметить, что необходимая категория находится в блоке «content». Осталось получить список товаров. Становится очевидно, что они содержатся в блоке «product-list». Такие характеристики товара как цена, ссылка, название можно получить через соответствующие им тэги и классы «price», «href», «name».

2.4.2 Занесение информации в базу данных

После сбора информации о товарах необходимо ее занести в базу данных при помощи библиотеки Doctrine. Для этого были созданы три класса: Brand, Model, Goods, хранящие марку, модели и сопутствующие товары. Далее через командную строку был сделан запрос на создание методов доступа к полям класса.

После этого в контроллере необходимо подключить классы для работы с ними, а также файл «parser.php», содержащий функцию для парсинга сайта. В одном из методов контроллера необходимо вызвать функцию Parse(), возвращающую массив из товаров. Но исходный массив не соответствует нашей модели базы данных, поэтому нужно извлечь нужные данные для каждой таблицы в одном из трех вложенных циклов, используя методы Set().

Так же нужно предусмотреть наличие записи в базе данных, поэтому перед записью необходимо послать запрос на поиск текущей записи.

После выполнения программы база данных наполнится следующими записями (см. рисунок 6, 7, 8).

+ Параметры

	brand_id	name
Изменить Копировать Удалить	1	BYD
Изменить Копировать Удалить	2	CHERY
Изменить Копировать Удалить	3	Chevrolet
Изменить Копировать Удалить	4	DAEWOO
Изменить Копировать Удалить	5	FAW
Изменить Копировать Удалить	6	GEELY
Изменить Копировать Удалить	7	GREAT WALL
Изменить Копировать Удалить	8	HAIMA
Изменить Копировать Удалить	9	HYUNDAI
Изменить Копировать Удалить	10	KIA
Изменить Копировать Удалить	11	LIFAN
Изменить Копировать Удалить	12	VORTEX

Рисунок 6 – таблица Brand

	model_id	name	brand_id
Изменить Копировать Удалить	1	F-3	1
Изменить Копировать Удалить	2	F3R	1
Изменить Копировать Удалить	3	Flyer	1
Изменить Копировать Удалить	4	Amulet	2
Изменить Копировать Удалить	5	Bonus	2
Изменить Копировать Удалить	6	Boo M11	2
Изменить Копировать Удалить	7	Boo M12	2
Изменить Копировать Удалить	8	Cross Eastar	2
Изменить Копировать Удалить	9	Fora	2
Изменить Копировать Удалить	10	IndiS	2
Изменить Копировать Удалить	11	Karry	2
Изменить Копировать Удалить	12	Kimo	2
Изменить Копировать Удалить	13	QQ	2
Изменить Копировать Удалить	14	QQ6	2
Изменить Копировать Удалить	15	Riich	2
Изменить Копировать Удалить	16	Tiggo	2

Рисунок 7 – таблица Model

	goods_id	name	price	link	model_id
Изменить Копировать Удалить	1	Ремень ГУР и кондиционера	350	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	2	Ремень ГУР и кондиционера 1.5-1.6	430	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	3	Стойка переднего стабилизатора	450	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	4	Амортизатор задний левый	3400	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	5	Амортизатор задний правый	3400	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	6	Амортизатор передний левый	1700	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	7	Амортизатор передний правый	2500	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	8	Бачок омывателя	620	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	9	Брызговик задний левый	150	http://an-auto.ru/index.php?route=product/product&...	1
Изменить Копировать Удалить	10	Брызговик задний правый	150	http://an-auto.ru/index.php?route=product/product&...	1

Рисунок 8 – таблица Goods

Заключение

В ходе работы реализовано приложение, получающее информацию о товарах с интернет магазинов и заносящее информацию о них в базу данных, и предоставляющее выборку, и отображение каталога товаров.

Список использованной литературы

1. Bacon, P., Angularjs, D. Web Application Development/ Peter Bacon, Darwin Angularjs M.:Книга по Требованию, 2013. - 372 с.
2. Bowler, T. Symfony 1.3 Web Application Development/ Tim Bowler. M.:Книга по Требованию , 2009. - 228 с.
3. Fuecks, H. The PHP Anthology: Object Oriented PHP Solutions, Volume II/ Harry Fuecks. – SitePoint Pty. Ltd., 2004. - 420 с.
4. Potencier, F. Practical symfony 1.2 for Doctrine - second edition/ Fabien Potencier. - Sensio SA, 2009. – 388 p.
5. Schlossnagle, G. Advanced PHP Programming: Developing Large-scale Web Applications With PHP 5 (Developer's Library)/ George Schlossnagle. M.:Харвест, 2008. - 700 с.
6. Sheltren, J. High Performance Drupal / J. Sheltren, N. Newton, N. Catchpole. – California: O'Reilly Media, Inc., 2013. – 263 p.
7. Vaswani ,V. XML and PHP/ Vikram Vaswani. – New Riders, 2002. – 384 p.
8. Аткинсон, Л. PHP 5 Библиотека профессионала/Леон Аткинсон. М.: Вильямс, 2005. - 944 с.
9. Горнаков, С.Г. Осваиваем популярные системы управления сайтом (CMS) / С.Г. Горнаков. – М.: ДМК Пресс, 2009. – 336 с.
10. Колисниченко Д. Профессиональное программирование на PHP/ Д. Колисниченко. М.:БХВ-Петербург, 2007. - 416 с.
11. Кузнецов М., Симдянов И. Объектно-ориентированное программирование на PHP/ М. Кузнецов, И. Симдянов. - М.: БХВ-Петербург, 2008. - 608 с.
12. Маркин А. В., Шкарин С. С. Основы Web-программирования на PHP/ А.В. Маркин, С.С. Шкарин. - Диалог-МИФИ, 2012. - 256 с.
13. Пейтон К., Меллер А. PHP 5 & MySQL 5/ К. Пейтон, А. Меллер. М.:Бином-Пресс, 2009. – 366 с.
14. Спикльмайр, С. Разработка Web-приложений и управление контентом /

С. Спикльмайр, К. Фридли, Дж. Спикльмайр, К. Брэнд. – М.: ДМК Пресс, 2003. - 464 с.

15. Холзнер, С. PHP в примерах/ С. Холзнер. М.:Бином-Пресс, 2007. – 352 с.