



**Компьютерная академия «ШАГ»
Одесский филиал
Кафедра Разработки программного обеспечения**

КУРСОВОЙ ПРОЕКТ ПО WEB-ПРОГРАММИРОВАНИЮ
“Книжный интернет магазин”

Студента группы ВПУ-1311

Ковалева Д.А.

Руководитель курсового проекта:

Полянский В.В.

Одесса 2016

АННОТАЦИЯ

Книжный интернет магазин «HELLO WORLD» предназначен для того чтобы предоставить покупателю удобный инструмент для выбора книги посвященной .NET технологиям. Целевым действием покупателя считается непосредственное оформление заказа с сайта. В ответ на который, будет отправлена посылка с наложенным платежом на тот адрес, который указан в заказе.

СОДЕРЖАНИЕ

Введение.....	4
1. Техническое задание.....	5
1.1 Постановка задачи	5
1.2 Требования к функциональным характеристикам системы	6
1.3 Список технологий и инструментальных средств.....	7
1.4 Требования к программным и техническим характеристикам системы.....	7
2. Выбор технологии для реализации проекта.....	8
3. Разработка структуры системы	10
3.1 Диаграмма компонентов.....	10
3.1 Диаграмма классов.....	11
3.2 Описание классов	13
4. Разработка алгоритмов функционирования системы	16
4.1 Диаграмма деятельности	16
5. Разработка базы данных для системы.....	17
5.1 Диаграмма базы данных	17
5.2 Описание таблиц базы данных	18
6. Разработка верстки сайта	19
7. Руководство пользователя.....	20
7.1 Основной режим работы	20
7.2 Режим администратора.....	26
Выводы.....	28
Список используемой литературы	29
Приложение 1. Листинг программы	30

ВВЕДЕНИЕ

Все те, кто занимался изучением .NET технологий знаю, как трудно найти нужную книгу, которая была бы написана простым и понятным языком. Хорошая книга должна прояснять сложные темы, а не запутывать. Но, к сожалению, найти хорошую книгу бывает, порой, очень трудной задачей. Для того чтобы не ошибиться с выбором, нужен книжный интернет магазин, где собраны самые лучшие и проверенные временем книги. Разработанный в ходе курсового проектирования книжный интернет магазин "HELLO WORLD" позволяет упростить задачу выбора книги. Сайт позволяет выбрать качественную и, самое важное, не дорогую книгу.

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Постановка задачи

В рамках курсовой работы по ASP.NET MVC необходимо разработать книжный интернет магазин, который позволял бы покупателю выбрать себе подходящую книгу по .NET технологии и ознакомиться с предложенным ассортиментом.

При этом приложение должно удовлетворять следующим требованиям:

- база данных должна содержать развернутое описание книги. Для каждой книги необходимо хранить (категорию, название, количество страниц, стоимость, описание);
- наличие возможности регистрации нового покупателя и входа
- сайт должен иметь корзину;
- корзину можно редактировать. Добавлять новые книги и удалять не нужные;
- по завершению покупки нужно указать адрес доставки и имя получателя;
- все новые заказы должны отправляться логисту;
- сайт должен содержать доступ к заполнению новых книг, категорий и пользователей;
- администратор может посмотреть историю покупок каждого покупателя;

Для хранения информации на сервере необходимо использовать базу данных, которая управляется СУБД MS SQL Server.

1.2 Требования к функциональным характеристикам системы

Функциональные характеристики системы представлены на диаграмме вариантов использования (рис. 1.1).

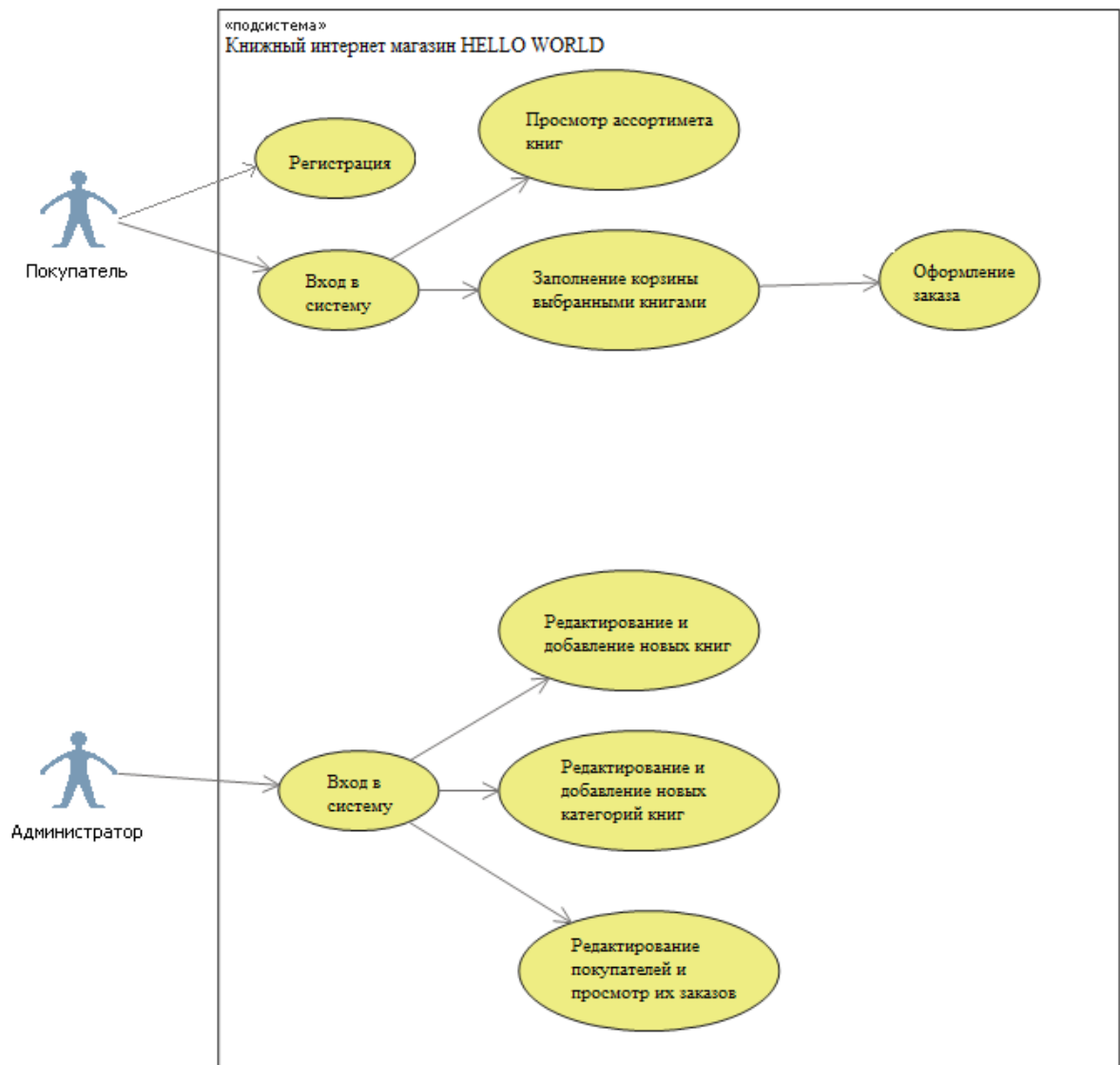


Рис. 1.1 - Диаграмма вариантов использования.

1.3 Список технологий и инструментальных средств

- SQL Server 2014 Express LocalDB;
- Microsoft Visual Studio 2013;
- Язык программирования C#;
- ASP.NET MVC 4.0;
- Ninject 3.2.2.0;
- Bootstrap 3.3.6;
- Entity Framework 6.1.3;
- jQuery 1.9.1

1.4 Требования к программным и техническим характеристикам системы

- Операционные системы – Windows 7;
- Платформа .NET FRAMEWORK 4.5.1;
- SQL Server 2014 Express LocalDB;
- IIS (Internet Information Services)
- Процессор с тактовой частотой 2 ГГц;
- Оперативная память 1 Гбайт;
- 1 Гбайт свободного места на жестком диске.

2. ВЫБОР ТЕХНОЛОГИИ ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТА

Для реализации проекта выбрана технология ASP.NET MVC Framework - фреймворк для создания веб-приложений, который реализует шаблон Model-view-controller.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления. Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Подобно всем приложениям .NET, приложения ASP.NET всегда компилируются. На самом деле выполнение кода на C# без предварительной компиляции просто невозможно.

Приложения ASP.NET в действительности проходят через два этапа компиляции.

На первом этапе написанный код на C# компилируется в код на промежуточном языке, который называется MSIL (Microsoft Intermediate Language — промежуточный язык Microsoft), или просто IL. Этот первый этап как раз и является одной из главных причин, по которым в .NET могут использоваться самые разные языки.

Второй этап компиляции происходит непосредственно перед фактическим выполнением страницы. На этом этапе код IL компилируется в код на низкоуровневом машинном языке. Называется этот этап оперативной (Just-In-Time — JIT) компиляцией и выглядит одинаково для всех приложений .NET (включая, например, Windows-приложения).

В ASP.NET также включены инструменты для выполнения предварительной компиляции, с помощью которых можно делать так, чтобы приложение компилировалось сразу же в машинный код прямо после его развертывания на производственном веб-сервере. Это позволяет избежать накладных расходов, связанных с выполнением первого этапа компиляции

при развертывании готового приложения (и исключить возможность подделки или изменения кода другими людьми).

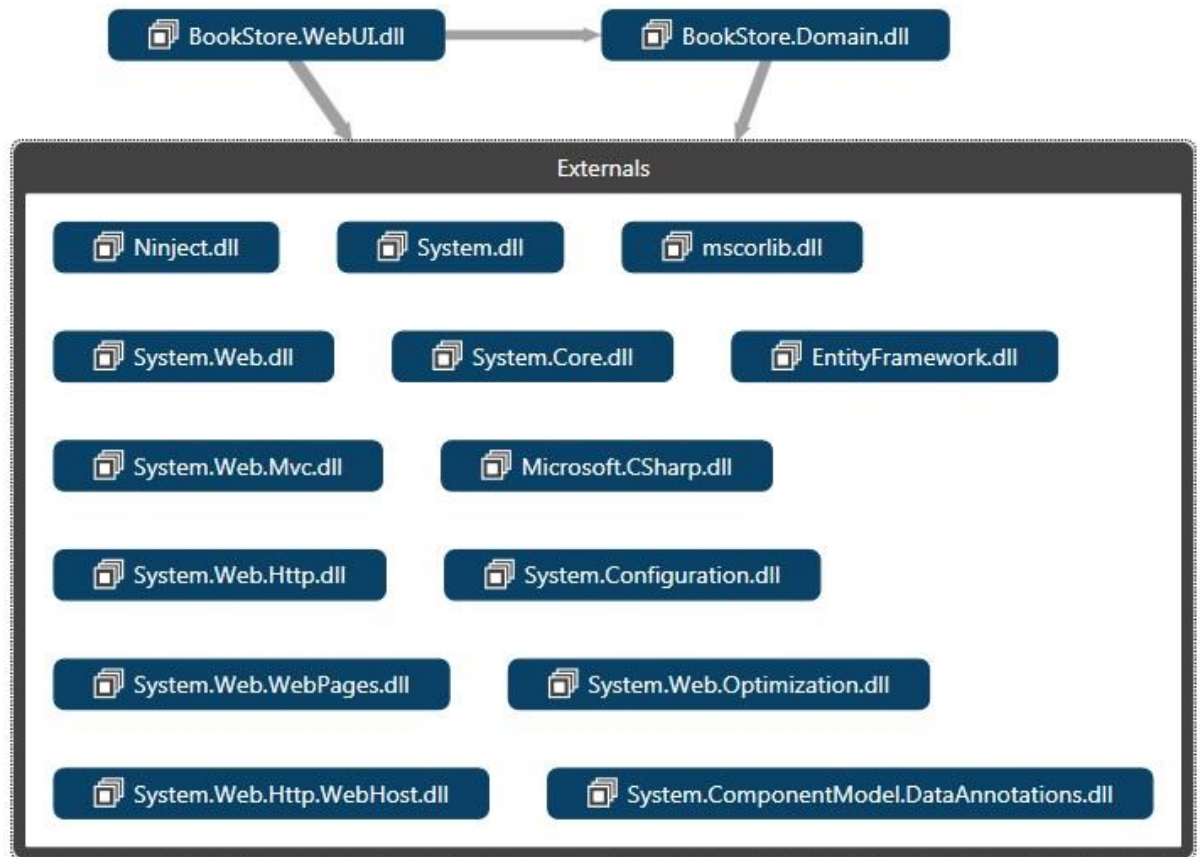
Для взаимодействия приложения с базой данной выбрана технология ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, которая является object-relational mapping (ORM) решением для .NET Framework от Microsoft.

Для того чтобы использовать объектно-ориентированный подход для взаимодействия с базой данных в приложении применяется язык интегрированных запросов Language Integrated Query (LINQ). Это проект Microsoft по добавлению синтаксиса языка запросов, напоминающего SQL, в языки программирования платформы .NET Framework.

Изначально поддерживая механизм запросов для коллекций объектов в памяти, реляционных баз данных и данных в формате XML, LINQ обладает расширяемой архитектурой, которая позволяет сторонним разработчикам реализовать доступ к их хранилищам данных через механизм LINQ. Для этого необходимо реализовать стандартные операторы запросов, используя методы расширения, или реализовать интерфейс IQueryable, позволяющий разбирать дерево выражения во время выполнения, транслируя его в свой язык запросов. В сообществе существует пример пользовательской реализации стандартных операторов запросов.

3. РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ

3.1. Диаграмма компонентов



3.2. Диаграмма классов

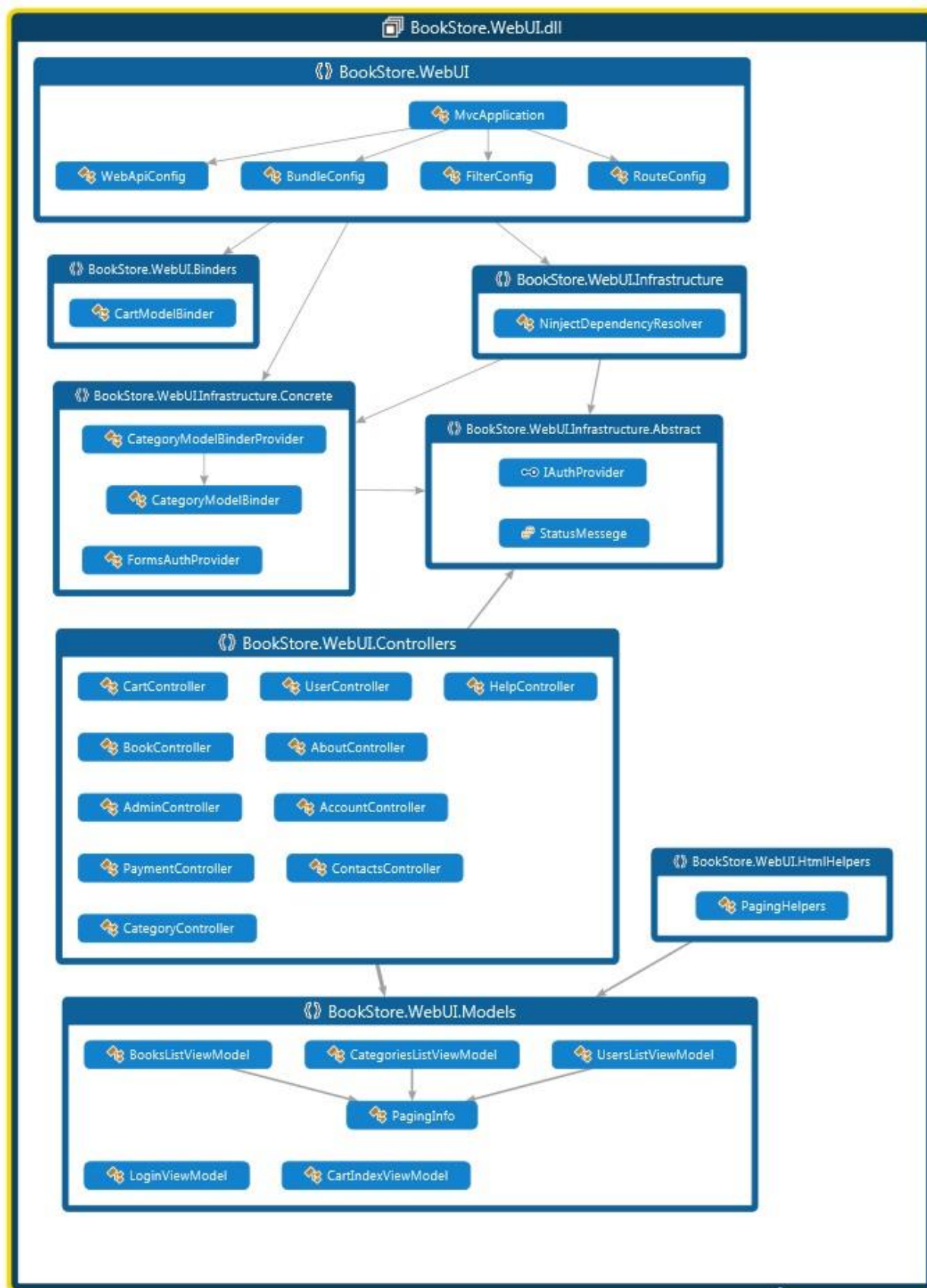


Рис. 3.1 – Классы пространства имен WebUI.

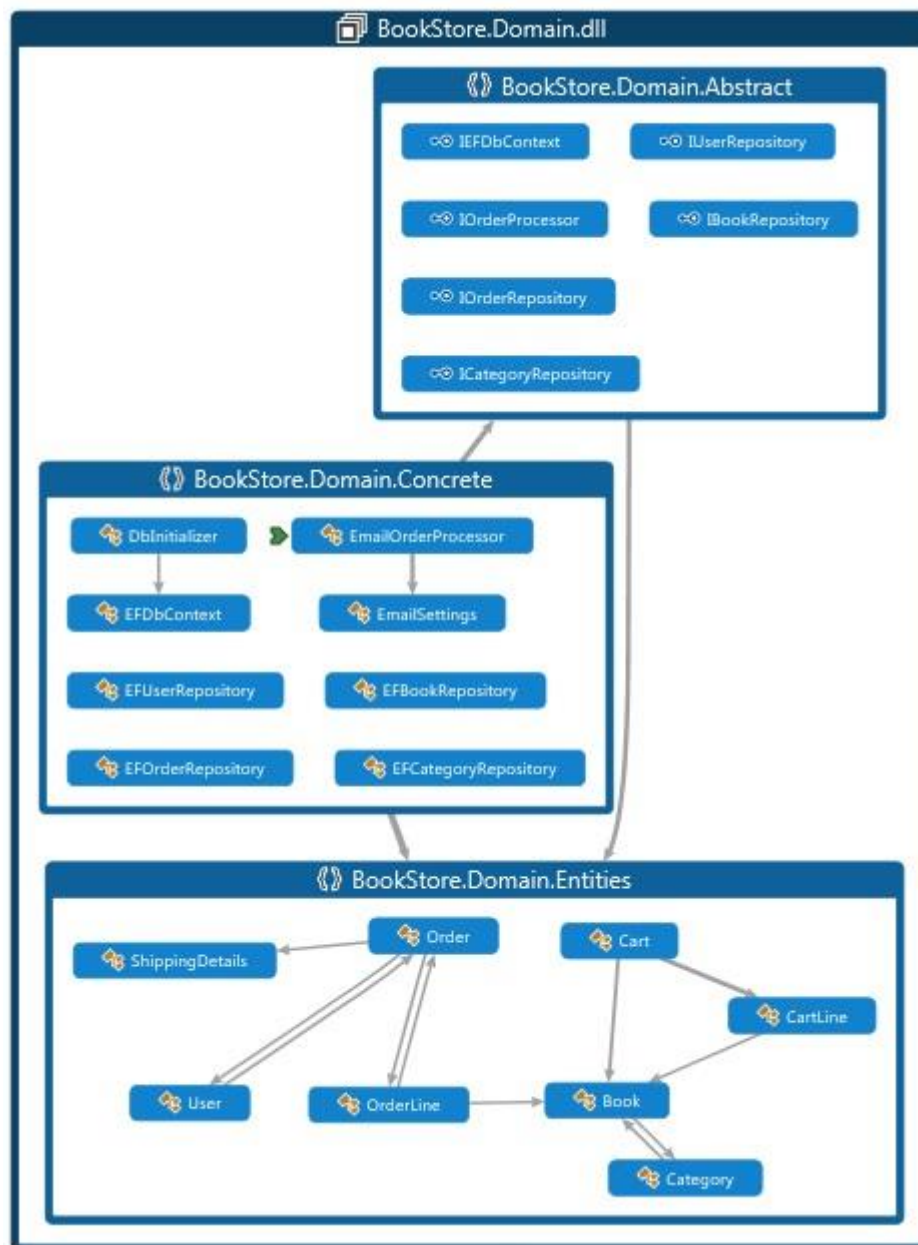


Рис. 3.2 – Классы пространства имен Domain.

3.3 Описание классов

Приложение включает в себя следующие классы:

MvcApplication – класс, который наследует `HttpApplication` и содержит инструкции для инициализации приложения.

WebApiConfig – класс, в котором находятся описание маршрутов Web API, однако может содержать и другие настройки конфигурации.

RouteConfig – класс определяющий маршруты.

FilterConfig - класс, который в методе `RegistreGlobalFilters` осуществляет регистрацию глобальных фильтров в коллекцию `filters`. Эта коллекция представляет собой объект `GlobalFilterCollection`, который передается в качестве параметра при вызове метода в файле `Global.asax`.

BundleConfig – регистрирует бандлы(наборы скриптов и стилей для более удобной передачи клиенту).

NinjectDependencyResolver – на основе интерфейса создает реализации, что позволяет разделить зоны ответственности.

CartModelBinder – привязчик модели `Cart`

CategoryModelBinder – привязчик модели `Category`

CartController – класс, предоставляет методы, реагирующие на HTTP-запросы (`Cart`), направляемые на веб-сайт MVC ASP.NET

AboutController – класс, предоставляет методы, реагирующие на HTTP-запросы (`About`), направляемые на веб-сайт MVC ASP.NET

AccountController – класс, предоставляет методы, реагирующие на HTTP-запросы (`Account`), направляемые на веб-сайт MVC ASP.NET

AdminController – класс, предоставляет методы, реагирующие на HTTP-запросы (`Admin`), направляемые на веб-сайт MVC ASP.NET

BookController – класс, предоставляет методы, реагирующие на HTTP-запросы (`Book`), направляемые на веб-сайт MVC ASP.NET

CategoryController – класс, предоставляет методы, реагирующие на HTTP-запросы (`Category`), направляемые на веб-сайт MVC ASP.NET

ContactsController – класс, предоставляет методы, реагирующие на HTTP-запросы (Contacts), направляемые на веб-сайт MVC ASP.NET

HelpController – класс, предоставляет методы, реагирующие на HTTP-запросы (Help), направляемые на веб-сайт MVC ASP.NET

PaymentController – класс, предоставляет методы, реагирующие на HTTP-запросы (Payment), направляемые на веб-сайт MVC ASP.NET

CategoryController – класс, предоставляет методы, реагирующие на HTTP-запросы (Category), направляемые на веб-сайт MVC ASP.NET

UserController – класс, предоставляет методы, реагирующие на HTTP-запросы (User), направляемые на веб-сайт MVC ASP.NET

PagingHelpers – класс, расширяющий HTTP хелпер. Добавляющий такой метод как PageLinks, формирующий элементы навигации по страницам сайта

CartItemViewModel – модель для передачи данных из контроллера CartController в его Views

CategoriesListViewModel - модель для передачи данных из контроллера CategoryController в его Views

LoginViewModel - модель для передачи данных из контроллера UserController в его Views

PagingInfo – содержит данные необходимые для навигации по страницам сайта

BooksListViewModel - модель для передачи данных из контроллера BookController в его Views

UsersListViewModel- модель для передачи данных из контроллера UserController в его Views

Category – класс содержит данные о категориях книг

Book – класс, содержащий данные о книге

User – класс, содержащий данные о пользователях сайта

User – класс, содержащий данные о пользователях сайта

Order– класс, содержащий данные о заказах сайта

OrderLine – класс, содержит данные о заказанной книге, ее количество и цену

ShippingDetails – класс, содержащий данные о адресе доставки заказа

Cart – класс, отвечающий за работу корзины сайта

EFBookRepository – класс, отвечающий за работу с книгами.
Получение, изменение и добавление новых книг в базу данных

EFCategoryRepository – класс, отвечающий за работу с категориями.
Получение, изменение и добавление категорий книг в базу данных

EFOrderRepository – класс, отвечающий за работу с заказами.
Получение и добавление новых заказов в базу данных

EFUserRepository – класс, отвечающий за работу с пользователями.
Получение, изменение и добавление новых пользователей в базу данных

EmailSettings – содержит настройки для отправки писем

EmailOrderProcessor – класс, реализующий рассылку данных о произведенных заказах лицам, которые будут отправлять заказанные книги покупателям.

EFDbContext - экземпляр контекста DbContext представляет сочетание шаблонов единицы работы и репозитория, которое может быть использовано для запроса от базы данных и группирования изменений, которые можно затем записать обратно в хранилище одним блоком. Используется в каждом репозитории.

4. РАЗРАБОТКА АЛГОРИТМОВ ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ

4.1 Диаграмма деятельности

Алгоритм функционирования системы представлен на диаграмме деятельности (рис. 4.1.).

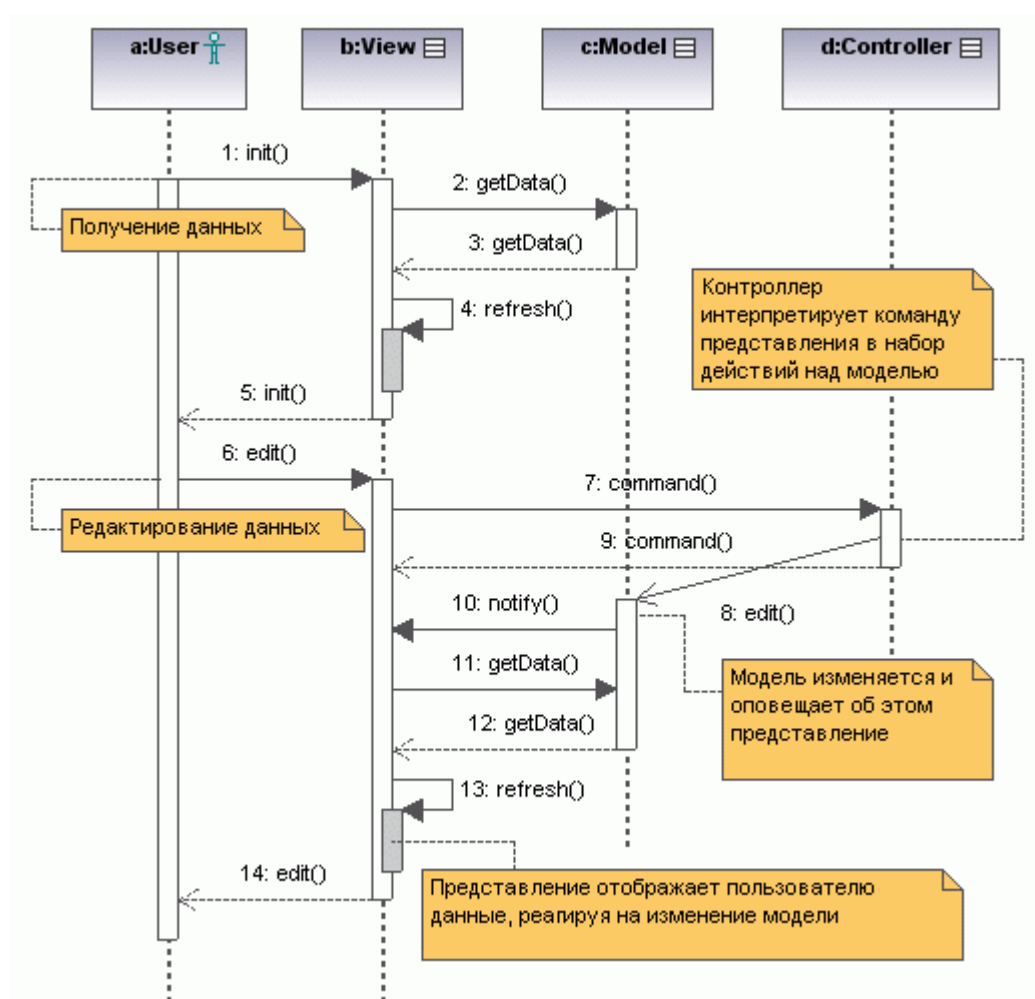


Рис. 4.1 - Диаграмма деятельности.

5. РАЗРАБОТКА БАЗЫ ДАННЫХ ДЛЯ СИСТЕМЫ

5.1 Диаграмма базы данных

База данных формируется при помощи Entity Framework. Диаграмма сформированной базы данных, представленная на рисунке 5.1, описывает структуру базы данных приложения.

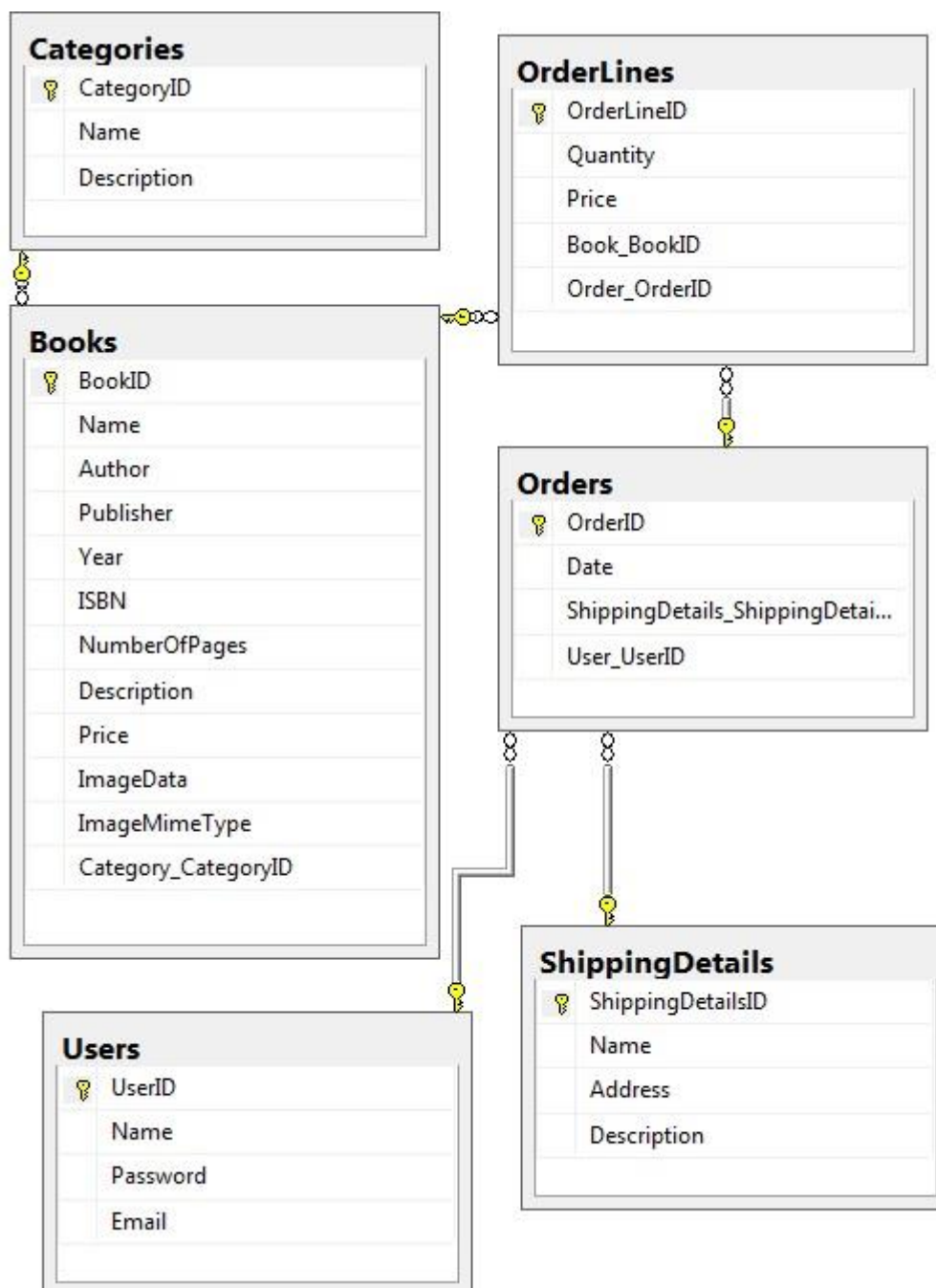


Рис. 5.1 - Диаграмма базы данных.

5.2 Описание таблиц базы данных

Category – таблица содержит данные о категориях книг

Book – таблица содержит данные о книге

User – таблица содержит данные о пользователях сайта

User – таблица содержит данные о пользователях сайта

Order– таблица содержит данные о заказах сайта

OrderLine – таблица содержит данные о заказанной книге, ее количество и цену

ShippingDetails – таблица содержит данные о адресе доставки заказа

6. РАЗРАБОТКА ВЕРСТКИ САЙТА

При разработке сайта предпочтение было отдано созданию верстки с использованием фреймворка Bootstrap.

Bootstrap — это фреймворк на основе HTML и CSS. Он содержит стили для основных элементов, которые применяются в верстке. Использование такого фреймворка значительно ускоряет процесс создания страниц. Стандартные стили легко менять, что обеспечивает гибкий и простой процесс создания макетов сайтов.

Основные преимущества Bootstrap:

1. Экономия времени — Bootstrap позволяет сэкономить время и усилия, используя шаблоны дизайна и классы, и сконцентрироваться на других разработках;
2. Высокая скорость — динамичные макеты Bootstrap масштабируются на разные устройства и разрешения экрана без каких-либо изменений в разметке;
3. Гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки. Дизайн и макеты веб-страниц согласуются друг с другом;
4. Простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с Bootstrap;
5. Совместимость с браузерами — Bootstrap совместим с Mozilla Firefox, Yandex Browser, Google Chrome, Safari, Internet Explorer и Opera;
6. Открытое программное обеспечение — особенность Bootstrap, которая предполагает удобство использования, посредством открытости исходных кодов и бесплатной загрузки.

7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

7.1 Основной режим работы.

При запуске сайта появится главная страница с перечнем всего ассортимента книг

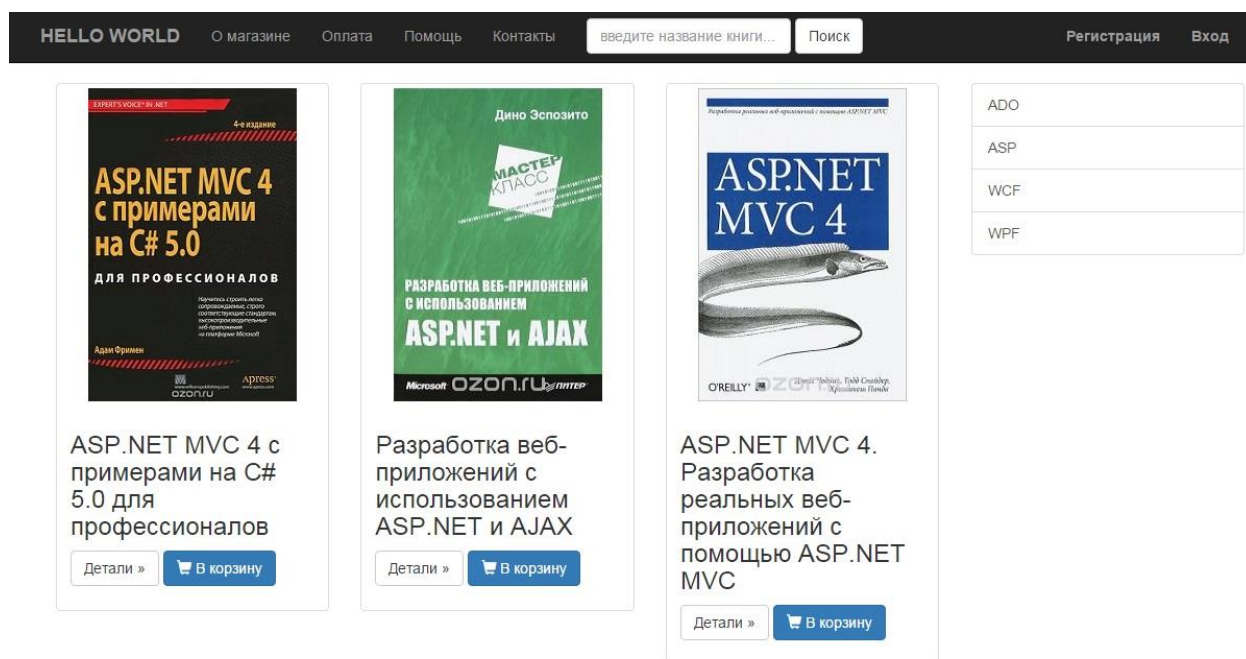


Рис 7.1 – главная страница сайта.

На главной панели расположены вкладки «О магазине», «Оплата», «Помощь», «Контакты». При нажатии на которые, открываются соответствующие страницы. В верхней правой части страницы расположено меню категорий книг, при нажатии на которую, открывается страница книг с соответствующей категории.

Книжный магазин "HELLO WORLD"

Интернет магазин "HELLO WORLD" предлагает книги, учебники по программированию и многое другое.

Покупать товары через интернет очень удобно, так как Вы можете ознакомиться с представленными книгами в интернет-магазине, а потом заказать или купить книги, которые вам понравились. На сегодняшний день книжный магазин предоставляет покупателям доставку книг Укрпочтой, перевозчиком «Новая почта» и самовывоз.

Книжный магазин "HELLO WORLD" - это всегда гибкая ценовая политика. Например, мы создаем особые условия для постоянных покупателей и для того, чтобы Вы, наш читатель, захотели приобрести книги именно в нашем книжном магазине. Специально для покупателей книжного магазина разработана программа лояльности, участвуя в которой можно покупать с постоянно увеличивающейся скидкой. Уже сегодня вы можете сделать покупку в нашем книжном магазине и не платить за ее доставку, скачать флаер и купить книгу со скидкой и это только начало...

Ассортимент книжного магазина формируется опытными товароведами, которые заказывают популярные и редкие книги, умело подбирают книги в рубрики новинки и распродажа.

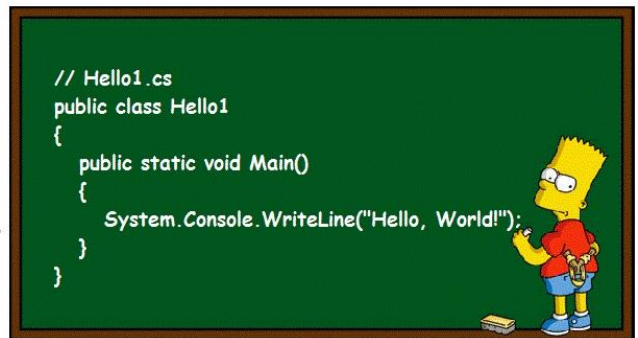


Рис 7.2 – Страница с описанием сайта.

Оплата

Оплатить сделанный заказ можно несколькими способами:

1. Оплата наличными в пункте самовывоза

Вы оплачиваете сделанный заказ наличными в пункте самовывоза. Адреса пунктов вы можете посмотреть [здесь](#).

2. Оплата наличными курьеру (Одесса)

Вы оплачиваете заказ курьеру при его получении.

3. Наложенный платеж (Новая почта)

Вы оплачиваете товар при получении в отделении Новой почты. Общая сумма включает в себя стоимость заказанных товаров + 25 грн доставка.

Если Вам не подходит не один из перечисленных вариантов оплаты, напишите об этом в комментариях при оформлении заказа. Наш оператор обязательно свяжется с Вами для согласования способа оплаты.



Компьютерная академия ШАГ

© 2016 Ковалев Денис. Группа ВПУ-1311

Рис 7.3 – Страница с описанием оплаты.

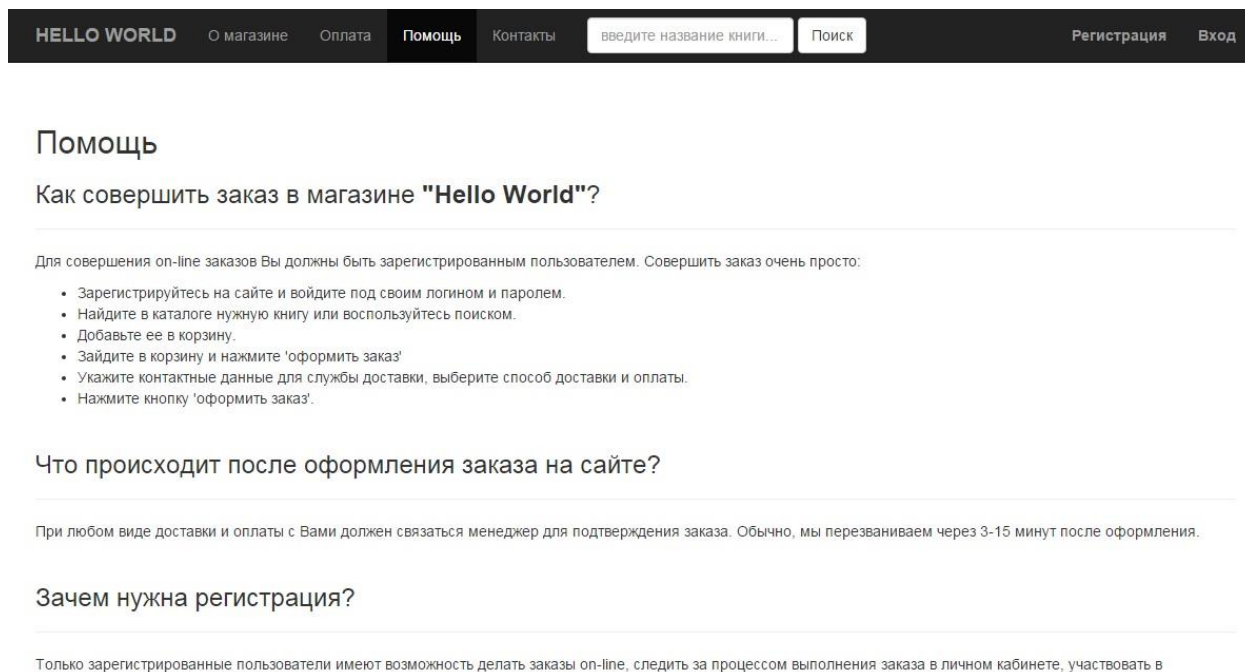


Рис 7.4 – Страница с описанием оформления заказа.

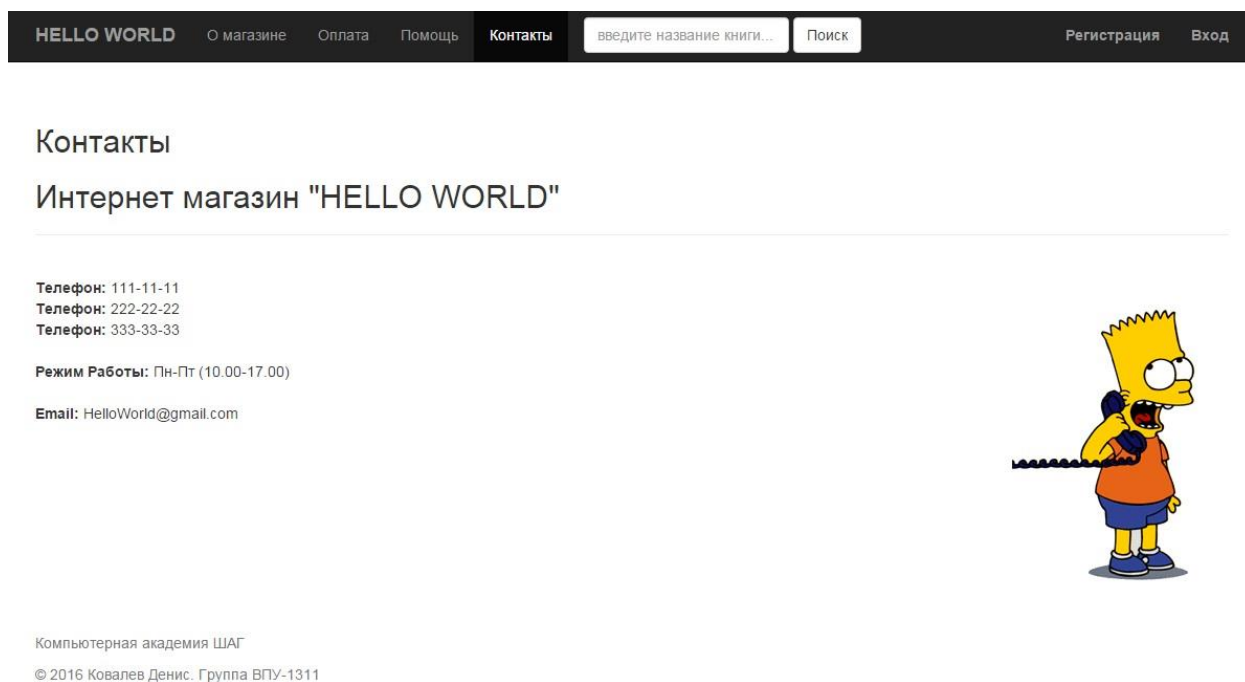


Рис 7.5 – Страница с контактами.

В верхней панели расположены команды **Регистрация** (открывает окно регистрации) и **Вход** (для входа на сайт под своим логином).

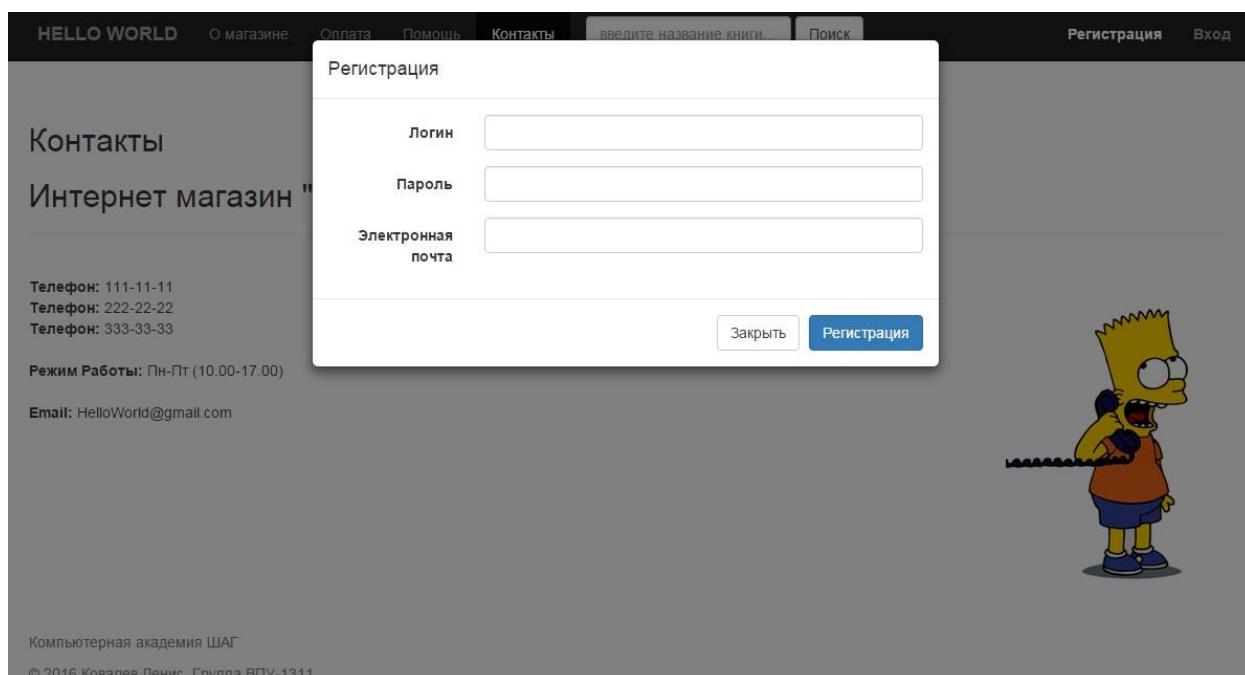


Рис 7.6 – Окно регистрации нового пользователя.

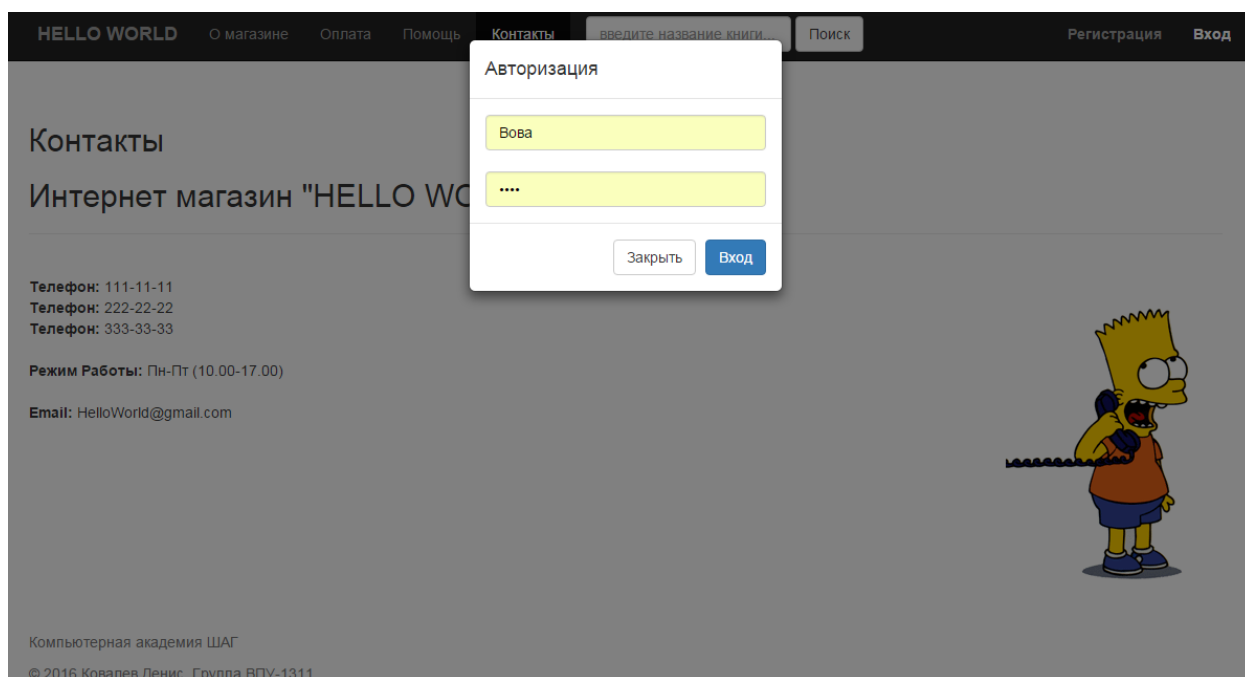


Рис 7.7 – Окно входа пользователя.

Напротив кнопки «поиск» расположен input, в который можно внести название или часть названия книги, которую требуется найти на сайте. В результате нажатия кнопки «Поиск» будет открыта страница с найденными книгами.

Напротив каждой книги есть две кнопки «Детали» и «В корзину»

HELLO WORLD

О магазине

Оплата

Помощь

Контакты

введите название книги...

Поиск

Вова

Корзина 0 грн.

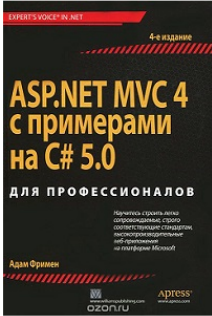
ADO

ASP

WCF

WPF

ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов



Автор	Адам Фримен
Издатель	Питер
ISBN	978-5-496-00657-6
Количество страниц	666
Год	2014
Цена	700,00

В корзину

Научитесь с помощью книги «ASP.NET MVC 4 Framework с примерами на C# 5.0 для профессионалов» строить легко сопровождаемые, строго соответствующие стандартам, высокопроизводительные веб-приложения на платформе Microsoft! ASP.NET MVC 4 представляет собой последнюю версию веб-платформы ASP.NET от Microsoft. Эта веб-платформа предлагает высокопроизводительную модель программирования, которая способствует построению более чистой кодовой архитектуры, поддерживает разработку через тестирование и обеспечивает повсеместную расширяемость в комбинации со всеми преимуществами ASP.NET. В четвертом издании книги «ASP.NET MVC 4 Framework с примерами на C# 5.0 для профессионалов» ключевые концепции архитектуры "модель-представление-контроллер" (MVC) не просто объясняются или обсуждаются в изоляции, но демонстрируются в действии. В вашем распоряжении — расширенное учебное руководство, позволяющее создать работающее веб-приложение электронного магазина, в котором сочетаются возможности ASP.NET MVC и новейшие средства

Рис 7.8 – Страница с деталями книги.

На странице с деталями книги можно узнать подробную информацию о книге. Таковую как, автор, издатель, количество страниц, год издания, описание книги и цену.

HELLO WORLD

О магазине

Оплата

Помощь

Контакты

введите название книги...

Поиск

Вова

Корзина 700 грн.

Ваша корзина

Количество	Название	Цена	Сумма	
1	ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов	700 грн.	700 грн.	Удалить
Всего:			700 грн.	

Продолжить покупки

Оформить заказ

Рис 7.9 – Страница с корзиной.

В корзине можно удалить выбранную книгу при нажатии на кнопку «Удалить» либо продолжить покупки, нажав кнопку «Продолжить покупки».

Если все необходимые книги выбраны, можно перейти к оформлению заказа, нажав кнопку «Оформить заказ».

HELLO WORLD О магазине Оплата Помощь Контакты Поиск Вова Корзина 700 грн.

Оформление заказа

Пожалуйста проверьте детали, и мы отправим ваши книги сегодня же!

ФИО получателя

Адрес

Комментарий к посылке

Рис 7.9 – Страница оформления заказа.

На странице оформления заказа нужно указать ФИО получателя посылки, адрес доставки и комментарий покупателя, в котором можно указать особые пожелания, такие как желаемая дата доставки или особые условие доставки.

7.2 Режим администратора.

Сайт содержит страницы для администратора, в которых можно редактировать базу данных. Для доступа к этим страницам необходимо ввести логин и пароль администратора в окне авторизации (рис 7.10). Логин администратора «Admin» и пароль «1111».

[Книги](#) [Категории](#) [Пользователи](#)

Авторизация

Пожалуйста введите логи и пароль администратора

UserName

admin

Password

....

Войти

Рис 7.10 – Страница авторизации администратора.

[Книги](#) [Категории](#) [Пользователи](#)

Название	Автор	Издатель	Год издания	ISBN	Кол-во страниц	Цена	Добавить новую книгу	
ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов	Адам Фримен	Питер	2014	978-5-496-00657-6	666	700,00	Редактировать	Удалить
Разработка веб-приложений с использованием ASP.NET и AJAX	Дино Эспозито	Питер	2012	978-0-7356-2621-8	400	190,00	Редактировать	Удалить
ASP.NET MVC 4. Разработка реальных веб-приложений с помощью ASP.NET MVC	Джесс Чедвик	Вильямс	2013	978-5-8459-1841-3	431	320,00	Редактировать	Удалить
ASP.NET MVC 3 Framework с примерами на C# для профессионалов	Адам Фримен	Вильямс	2011	978-5-8459-1758-4	672	710,00	Редактировать	Удалить
ASP.NET. Сборник рецептов (+ CD-ROM)	Павел Агуров	БХВ-Петербург	2010	978-5-9775-0521-5	528	100,00	Редактировать	Удалить
Самоучитель ASP.NET	Игорь Шапошников	БХВ-Петербург	2010		0	20,00	Редактировать	Удалить
Wcf 4.5 Multi-Layer Services Development with Entity Framework	Mike Liu		2012	9781849687669	394	2300,00	Редактировать	Удалить

Рис 7.11 – Страница редактирования книг.

Здесь можно изменять, удалять и добавлять новые книги, категории книг и пользователей. При изменении любого объекта производится валидация введенных данных. В случае неверно введенных данных, будет подсвечены неверные поля, которые требуется исправить, для того чтобы можно было сохранить данные.

The screenshot shows a web interface for editing books. At the top, there are three tabs: 'Книги' (Books), 'Категории' (Categories), and 'Пользователи' (Users). Below the tabs, the page title is 'Редактирование книги:' followed by '— ID 2'. A red error message box is displayed, stating 'ОШИБКИ:' and 'Пожалуйста введите название книги'. Below the error message, there are five input fields for book details: 'Название' (Name), 'Категория' (Category), 'Автор' (Author), 'Издатель' (Publisher), and 'Год' (Year). The 'Название' field is empty and has a red border, indicating it is the source of the error. The 'Категория' field is a dropdown menu with 'ASP' selected. The 'Автор' field contains 'Дино Эспозито', the 'Издатель' field contains 'Питер', and the 'Год' field contains '2012'.

Книги	Категории	Пользователи
-------	-----------	--------------

Редактирование книги:
— ID 2

ОШИБКИ:

- Пожалуйста введите название книги

Название	<input type="text"/>
Категория	ASP
Автор	Дино Эспозито
Издатель	Питер
Год	2012

Рис 7.12 – Валидация данных

ВЫВОДЫ

В ходе работы над курсовым проектом был создан книжный интернет магазин «HELLO WORLD», который предоставляет широчайший ассортимент книг посвященным .NET технологиям

При разработке сайта были использованы платформа .NET Framework, ASP.NET MVC , язык программирования C#, технология взаимодействия с БД Entity Framework.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Адам Фримен - С ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов
2. Рихтер Дж. - CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#
3. Кристиан Нейгел - C# 2008 и платформа NET 3.5 для профессионалов, И.Д. Вильямс
4. Э. Троельсен - Язык программирования C# 2008 и платформа .NET 3.5
5. <http://metanit.com/sharp/mvc.php>
6. http://professorweb.ru/my/ASP_NET/base/level1/aspnet_info.php
7. http://www.w3schools.com/aspnet/mvc_intro.asp /

Приложение 1. Листинг программы.

Abstract.cs

```
using System.Linq;
using BookStore.Domain.Entities;
using System.Data.Entity;

namespace BookStore.Domain.Abstract
{
    public interface IBookRepository
    {
        IQueryable<Book> Books { get; }
        void SaveBook(Book book);
        Book DeleteBook(int BookID);
    }

    public interface ICategoryRepository
    {
        IQueryable<Category> Categories { get; }
        void SaveCategory(Category category);
        Category DeleteCategory(int CategoryID);
        Category GetByName(string name);
    }

    public interface IUserRepository
    {
        IQueryable<User> Users { get; }
        void SaveUser(User user);
        User DeleteUser(int UserID);
        User GetByName(string name);
    }

    public interface IOrderRepository
    {
        IQueryable<Order> Orders { get; }
        void SaveOrder(Order order);
        IQueryable<Order> GetByUser(string userName);
    }

    public interface IOrderProcessor
    {
        void ProcessOrder(Cart cart, ShippingDetails shippingDetails, User user);
    }

    public interface IEFDbContext
    {
        DbSet<Book> Books { get; set; }
        DbSet<Category> Categories { get; set; }
        DbSet<User> Users { get; set; }
        DbSet<Order> Orders { get; set; }
        DbSet<OrderLine> OrderLines { get; set; }
        DbSet<ShippingDetails> ShippingDetailss { get; set; }

        void SaveChangesContext();
    }
}
```

EFBookRepository.cs

```
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using System.Linq;
using System.Web.Mvc;

namespace BookStore.Domain.Concrete
{
    public class EFBookRepository : IBookRepository
    {
        private IEFDbContext context = DependencyResolver.Current.GetService<IEFDbContext>();

        public IQueryable<Book> Books
        {
            get { return context.Books; }
        }
    }
}
```

```

    }

    public void SaveBook(Book book)
    {
        if (book.BookID == 0)
        {
            book.Category = context.Categories.Find(book.Category.CategoryID);
            context.Books.Add(book);
        }
        else
        {
            Book dbEntry = context.Books.Find(book.BookID);
            if (dbEntry != null)
            {
                dbEntry.Name      = book.Name;
                dbEntry.Author    = book.Author;
                dbEntry.ISBN      = book.ISBN;
                dbEntry.NumberOfPages = book.NumberOfPages;
                dbEntry.Publisher = book.Publisher;
                dbEntry.Year      = book.Year;
                dbEntry.Description = book.Description;
                dbEntry.Price     = book.Price;
                dbEntry.Category   = context.Categories.Find(book.Category.CategoryID) ?? dbEntry.Category;
                dbEntry.ImageData  = book.ImageData;
                dbEntry.ImageMimeType = book.ImageMimeType;
            }
            context.SaveChangesContext();
        }
    }

    public Book DeleteBook(int BookID)
    {
        Book dbEntry = context.Books.Find(BookID);
        if (dbEntry != null)
        {
            context.Books.Remove(dbEntry);
            context.SaveChangesContext();
        }
        return dbEntry;
    }
}
}

```

EFCategoryRepository.cs

```

using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using System.Linq;
using System.Web.Mvc;

namespace BookStore.Domain.Concrete
{
    public class EFCategoryRepository : ICategoryRepository
    {
        private IEFDbContext context = DependencyResolver.Current.GetService<IEFDbContext>();

        public IQueryable<Category> Categories
        {
            get { return context.Categories; }
        }

        public void SaveCategory(Category category)
        {
            if (category.CategoryID == 0)
            {
                context.Categories.Add(category);
            }
            else
            {
                Category dbEntry = context.Categories.Find(category.CategoryID);
                if (dbEntry != null)
                {
                    dbEntry.Name = category.Name;
                    dbEntry.Description = category.Description;
                }
            }
        }
    }
}

```

```

        context.SaveChangesContext();
    }

    public Category DeleteCategory(int CategoryID)
    {
        Category dbEntry = context.Categories.Find(CategoryID);
        if (dbEntry != null)
        {
            context.Categories.Remove(dbEntry);
            context.SaveChangesContext();
        }
        return dbEntry;
    }

    public Category GetByName(string name)
    {
        return context.Categories.FirstOrDefault(c => c.Name == name);
    }
}

```

EFDbContext.cs

```

using BookStore.Domain.Entities;
using System.Data.Entity;
using BookStore.Domain.Abstract;

namespace BookStore.Domain.Concrete
{
    public class EFDbContext : DbContext, IEFDbContext
    {
        public DbSet<Book> Books { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<User> Users { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderLine> OrderLines { get; set; }
        public DbSet<ShippingDetails> ShippingDetailss { get; set; }
        public void SaveChangesContext()
        {
            this.SaveChanges();
        }
    }
}

```

EFOrderRepository.cs

```

using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using System.Linq;
using System.Web.Mvc;
using System.Collections.Generic;

namespace BookStore.Domain.Concrete
{
    public class EFOrderRepository : IOrderRepository
    {
        private IEFDbContext context = DependencyResolver.Current.GetService<IEFDbContext>();

        public IQueryable<Order> Orders
        {
            get { return context.Orders; }
        }

        public void SaveOrder(Order order)
        {
            ShippingDetails shippingDetail = null;
            if (order.ShippingDetails != null && order.ShippingDetails.ShippingDetailsID == 0)
            {
                shippingDetail = new ShippingDetails
                {
                    Name = order.ShippingDetails.Name,
                    Address = order.ShippingDetails.Address,
                    Description = order.ShippingDetails.Description
                };
                context.ShippingDetailss.Add(shippingDetail);
                context.SaveChangesContext();
            }
        }
    }
}

```



```

    }
    else
    {
        shippingDetail = context.ShippingDetailss.Find(shippingDetail.ShippingDetailsID);
    }

    User user = (order.User == null ? null : context.Users.Find(order.User.UserID));
    List<OrderLine> OrderLines = order.OrderLines.ToList();

    //сохраним сам заказ
    if (order.OrderID == 0)
    {
        order.ShippingDetails = shippingDetail;
        order.User = user;
        order.OrderLines = null;
        context.Orders.Add(order);
    }
    else
    {
        Order dbEntry = context.Orders.Find(order.OrderID);
        if (dbEntry != null)
        {
            dbEntry.Date = order.Date;
            dbEntry.ShippingDetails = shippingDetail;
            dbEntry.User = user;
        }
    }
    context.SaveChangesContext();

    //Добавим строки заказа
    foreach (OrderLine line in OrderLines)
    {
        OrderLine newOrderLine = new OrderLine {
            Book = context.Books.Find(line.Book.BookID),
            Price = line.Price,
            Quantity = line.Quantity,
            Order = context.Orders.Find(order.OrderID)
        };
        context.OrderLines.Add(newOrderLine);
    }
    context.SaveChangesContext();
}

public IQueryable<Order> GetByUser(string userName)
{
    return context.Orders.Where(c => c.User.Name == userName);
}

}
}

```

EFUserRepository.cs

```

using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using System.Linq;
using System.Web.Mvc;

namespace BookStore.Domain.Concrete
{
    public class EFUserRepository : IUserRepository
    {
        private IEFDbContext context = DependencyResolver.Current.GetService<IEFDbContext>();

        public IQueryable<User> Users
        {
            get { return context.Users; }
        }

        public void SaveUser(User user)
        {
            if (user.UserID == 0)
            {
                context.Users.Add(user);
            }
            else
            {

```

```

    {
        User dbEntry = context.Users.Find(user.UserID);
        if (dbEntry != null)
        {
            dbEntry.Name = user.Name;
            dbEntry.Password = user.Password;
            dbEntry.Email = user.Email;
        }
    }
    context.SaveChangesContext();
}

public User DeleteUser(int UserID)
{
    User dbEntry = context.Users.Find(UserID);
    if (dbEntry != null)
    {
        context.Users.Remove(dbEntry);
        context.SaveChangesContext();
    }
    return dbEntry;
}

public User GetByName(string name)
{
    return context.Users.FirstOrDefault(c => c.Name == name);
}
}

```

EmailOrderProcessor.cs

```

using System.Net.Mail;
using System.Text;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using System.Net;
using System.Web.Mvc;
using System;
using System.Web;
using System.Net.Mime;

namespace BookStore.Domain.Concrete
{
    public class EmailSettings
    {
        public string MailToAddress ";
        public string MailFromAddress
        public bool UseSsl = true;
        public string Username
        public string Password
        public string ServerName;
        public int ServerPort;
        public bool WriteAsFile = true;
        public string FileLocation;
    }

    public class EmailOrderProcessor : IOrderProcessor
    {
        private EmailSettings emailSettings;
        private IOrderRepository repository = DependencyResolver.Current.GetService<IOrderRepository>();

        public EmailOrderProcessor(EmailSettings settings)
        {
            emailSettings = settings;
        }

        public void ProcessOrder(Cart cart, ShippingDetails shippingInfo, User user)
        {
            //Записать заказ в БД
            SaveOrder(cart, shippingInfo, user);

            //Отправить админу письмо с заказом и деталями доставки
            SendOrdersToAdminMail(cart, shippingInfo);

            //отправим письмо клиенту
            SendMailToClient(cart, shippingInfo, user);
        }
    }
}

```

```

    }

    private void SendMailToClient(Cart cart, ShippingDetails shippingInfo, User user)
    {
        if (user == null || user.Email == "")
            return;

        using (var smtpClient = new SmtpClient())
        {
            smtpClient.EnableSsl = emailSettings.UseSsl;
            smtpClient.Host = emailSettings.ServerName;
            smtpClient.Port = emailSettings.ServerPort;
            smtpClient.UseDefaultCredentials = false;
            smtpClient.Credentials = new NetworkCredential(emailSettings.Username, emailSettings.Password);

            if (emailSettings.WriteAsFile)
            {
                smtpClient.DeliveryMethod = SmtpDeliveryMethod.SpecifiedPickupDirectory;
                smtpClient.PickupDirectoryLocation = emailSettings.FileLocation;
                smtpClient.EnableSsl = false;
            }

            StringBuilder body = GetBodyMail(cart, shippingInfo);

            MailMessage mailMessage = new MailMessage(
                emailSettings.MailFromAddress, // From
                user.Email, // To
                "Новый заказ!", // Subject
                body.ToString()); // Body

            mailMessage.BodyEncoding = Encoding.GetEncoding(1251);
            mailMessage.SubjectEncoding = Encoding.GetEncoding(1251);

            try { smtpClient.Send(mailMessage); }
            catch (Exception exp) {}
        }
    }

    private void SaveOrder(Cart cart, ShippingDetails shippingInfo, User user)
    {
        Order order = new Order();
        order.Date = DateTime.Now;
        order.User = user;
        order.ShippingDetails = shippingInfo;
        foreach (var item in cart.Lines)
        {
            order.OrderLines.Add(new OrderLine { Book = item.Book, Quantity = item.Quantity, Price = item.Book.Price });
        }

        repository.SaveOrder(order);
    }

    private void SendOrdersToAdminMail(Cart cart, ShippingDetails shippingInfo)
    {
        using (var smtpClient = new SmtpClient())
        {
            smtpClient.EnableSsl = emailSettings.UseSsl;
            smtpClient.Host = emailSettings.ServerName;
            smtpClient.Port = emailSettings.ServerPort;
            smtpClient.UseDefaultCredentials = false;
            smtpClient.Credentials = new NetworkCredential(emailSettings.Username, emailSettings.Password);
            smtpClient.EnableSsl = true;

            if (emailSettings.WriteAsFile)
            {
                smtpClient.DeliveryMethod = SmtpDeliveryMethod.SpecifiedPickupDirectory;
                smtpClient.PickupDirectoryLocation = emailSettings.FileLocation;
                smtpClient.EnableSsl = false;
            }

            StringBuilder body = GetBodyMail(cart, shippingInfo);

            MailMessage mailMessage = new MailMessage(
                emailSettings.MailFromAddress, // From
                emailSettings.MailToAddress, // To
                "Новый заказ!", // Subject

```

```

        body.ToString()); // Body
        mailMessage.BodyEncoding = Encoding.GetEncoding("Windows-1254");
        mailMessage.SubjectEncoding = Encoding.GetEncoding("Windows-1254");

        try { smtpClient.Send(mailMessage); }
        catch (Exception exp) { }

    }
}

private StringBuilder GetBodyMail(Cart cart, ShippingDetails shippingInfo)
{
    StringBuilder body = new StringBuilder()
        .AppendLine("Был оформлен новый заказ")
        .AppendLine("----")
        .AppendLine("Книги:");

    foreach (var line in cart.Lines)
    {
        var subtotal = line.Book.Price * line.Quantity;
        body.AppendFormat("{0} x {1} (итого: {2:c})", line.Quantity, line.Book.Name, subtotal);
    }

    body.AppendFormat("Сумма заказа: {0:c}", cart.ComputeTotalValue())
        .AppendLine("----")
        .AppendLine("Адрес отправки:")
        .AppendLine(shippingInfo.Name)
        .AppendLine(shippingInfo.Address)
        .AppendLine(shippingInfo.Description ?? "")
        .AppendLine("----");

    return body;
}
}
}

```

Cart.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BookStore.Domain.Entities
{
    public class Cart
    {
        private List<CartLine> lineCollection = new List<CartLine>();
        public void AddItem(Book book, int quantity)
        {
            CartLine line = lineCollection
                .Where(p => p.Book.BookID == book.BookID)
                .FirstOrDefault();
            if (line == null)
            {
                lineCollection.Add(new CartLine
                {
                    Book = book,
                    Quantity = quantity
                });
            }
            else
            {
                line.Quantity += quantity;
            }
        }
        public void RemoveLine(Book book)
        {
            lineCollection.RemoveAll(l => l.Book.BookID == book.BookID);
        }
        public decimal ComputeTotalValue()
        {
            return lineCollection.Sum(e => e.Book.Price * e.Quantity);
        }
        public void Clear()
        {
        }
    }
}

```

```

        lineCollection.Clear();
    }
    public IEnumerable<CartLine> Lines
    {
        get { return lineCollection; }
    }
}
public class CartLine
{
    public Book Book { get; set; }
    public int Quantity { get; set; }
}
}

```

Entities.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;

namespace BookStore.Domain.Entities
{
    public class Category
    {
        public Category()
        {
            Books = new List<Book>();
        }

        [HiddenInput(DisplayValue = false)]
        public int CategoryID { get; set; }

        [Required(ErrorMessage = "Пожалуйста введите название категории")]
        [Display(Name = "Название")]
        public string Name { get; set; }

        [DataType(DataType.MultilineText)]
        [Required(ErrorMessage = "Пожалуйста введите описание категории")]
        [Display(Name = "Описание")]
        public string Description { get; set; }

        public virtual ICollection<Book> Books { get; set; }
    }

    public class Book
    {
        [HiddenInput(DisplayValue = false)]
        public int BookID { get; set; }

        [Required(ErrorMessage = "Пожалуйста укажите категорию")]
        [Display(Name = "Категория")]
        public virtual Category Category { get; set; }

        [Required(ErrorMessage = "Пожалуйста введите название книги")]
        [Display(Name = "Название")]
        public string Name { get; set; }

        [Display(Name = "Автор")]
        public string Author { get; set; }

        [Display(Name = "Издатель")]
        public string Publisher { get; set; }

        [Display(Name = "Год")]
        public int Year { get; set; }
        public string ISBN { get; set; }

        [Display(Name = "Количество страниц")]
        public int NumberOfPages { get; set; }

        [DataType(DataType.MultilineText)]
    }
}

```

```

[Required(ErrorMessage = "Пожалуйста введите описание книги")]
[Display(Name = "Описание")]
public string Description { get; set; }

[Required]
[Range(0.01, double.MaxValue, ErrorMessage = "Пожалуйста введите положительную цену")]
[Display(Name = "Цена")]
public decimal Price { get; set; }

public byte[] ImageData { get; set; }

[HiddenInput(DisplayValue = false)]
public string ImageMimeType { get; set; }

}

public class User
{
    public User()
    {
        Orders = new List<Order>();
    }

    [HiddenInput(DisplayValue = false)]
    public int UserID { get; set; }

    [Required(ErrorMessage = "Пожалуйста введите логин")]
    [Display(Name = "Логин")]
    public string Name { get; set; }

    [Required(ErrorMessage = "Пожалуйста введите пароль")]
    [Display(Name = "Пароль")]
    public string Password { get; set; }

    [Display(Name = "Электронная почта")]
    [RegularExpression("[\\._-a-z0-9]+@[a-z0-9][\\._-a-z]{2,6}", ErrorMessage="Неправильный формат Email")]
    public string Email { get; set; }

    public virtual ICollection<Order> Orders { get; set; }
}

public class Order
{
    public Order() {
        OrderLines = new List<OrderLine>();
    }

    [HiddenInput(DisplayValue = false)]
    public int OrderID { get; set; }

    [Display(Name = "Дата")]
    [DisplayFormat(DataFormatString = "{0:dd MMM yyyy}")]
    public DateTime Date { get; set; }

    [Display(Name = "Покупатель")]
    public virtual User User { get; set; }

    public virtual ShippingDetails ShippingDetails { get; set; }
    public virtual ICollection<OrderLine> OrderLines { get; set; }
}

public class OrderLine
{
    [HiddenInput(DisplayValue = false)]
    public int OrderLineID { get; set; }

    public virtual Order Order { get; set; }

    public virtual Book Book { get; set; }

    public int Quantity { get; set; }

    public decimal Price { get; set; }
}

public class ShippingDetails
{

```

```

[HiddenInput(DisplayValue = false)]
public int ShippingDetailsID { get; set; }

[Required(ErrorMessage = "Пожалуйста введите полное имя получателя")]
[Display(Name = "ФИО получателя")]
public string Name { get; set; }

[Required(ErrorMessage = "Пожалуйста введите адрес")]
[Display(Name = "Адрес")]
public string Address { get; set; }

[DataType(DataType.MultilineText)]
[Display(Name = "Комментарий к посылке")]
public string Description { get; set; }

    }
}

```

RouteConfig.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace BookStore.WebUI
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(null,
                "",
                new
                {
                    controller = "Book",
                    action = "List",
                    category = (string)null,
                    page = 1
                });

            routes.MapRoute(null,
                "Page{page}",
                new { controller = "Book", action = "List", category = (string)null },
                new { page = @"\d+" }
            );

            routes.MapRoute(null,
                "Admin/{action}/Page{page}",
                new { controller = "Admin", page = 1 },
                new { page = @"\d+" }
            );

            routes.MapRoute(null,
                "Admin",
                new { controller = "Admin", action = "Books" }
            );

            routes.MapRoute(null,
                "{category}",
                new { controller = "Book", action = "List", page = 1 }
            );

            routes.MapRoute(null,
                "{category}/Page{page}",
                new { controller = "Book", action = "List" },
                new { page = @"\d+" }
            );

            routes.MapRoute(null, "{controller}/{action}");
        }
    }
}

```

CartModelBinder.cs

```
using System;
using System.Web.Mvc;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Binders
{
    public class CartModelBinder : IModelBinder
    {
        private const string sessionKey = "Cart";

        public object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
        {
            // получим корзину из сессии
            Cart cart = (Cart)controllerContext.HttpContext.Session[sessionKey];

            // создадим новую корзину если ее нет
            if (cart == null)
            {
                cart = new Cart();
                controllerContext.HttpContext.Session[sessionKey] = cart;
            }

            return cart;
        }
    }
}
```

AboutController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.WebUI.Controllers
{
    public class AboutController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.ActiveTab = "About";
            return View();
        }
    }
}
```

AccountController.cs

```
using System.Web.Mvc;
using BookStore.WebUI.Infrastructure.Abstract;
using BookStore.WebUI.Models;

namespace BookStore.WebUI.Controllers
{
    public class AccountController : Controller
    {
        IAuthProvider authProvider;
        public AccountController(IAuthProvider auth)
        {
            authProvider = auth;
        }

        public ViewResult Login()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Login(LoginViewModel model, string returnUrl)
        {

```



```

        if (ModelState.IsValid)
        {
            if (authProvider.Authenticate(model.UserName, model.Password))
            {
                return Redirect(returnUrl ?? Url.Action("Books", "Admin"));
            }
            else
            {
                ModelState.AddModelError("", "Неправильный логин или пароль");
                ViewBag.ErrorValidation = true;
                return View();
            }
        }
        else
        {
            ViewBag.ErrorValidation = true;
            return View();
        }
    }
}
}

```

AdminController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using BookStore.WebUI.Infrastructure.Abstract;
using BookStore.WebUI.Models;
using System.Configuration;

namespace BookStore.WebUI.Controllers
{
    [Authorize]
    public class AdminController : Controller
    {
        private IBookRepository repositoryBook;
        private ICategoryRepository repositoryCategory;
        private IUserRepository repositoryUser;
        private int PageSize = Convert.ToInt32(ConfigurationManager.AppSettings["NumberOfBooksOnThePage"]); //9

        public AdminController(IBookRepository repoBook, ICategoryRepository repoCategory, IUserRepository repoUser)
        {
            repositoryBook = repoBook;
            repositoryCategory = repoCategory;
            repositoryUser = repoUser;
        }

        public ActionResult Books(int page = 1)
        {
            ViewBag.SelectedAdminTab = "Books";
            BooksListViewModel model = new BooksListViewModel
            {
                Books = repositoryBook.Books
                    .OrderBy(p => p.BookID)
                    .Skip((page - 1) * PageSize)
                    .Take(PageSize),
                PagingInfo = new PagingInfo
                {
                    CurrentPage = page,
                    ItemsPerPage = PageSize,
                    TotalItems = repositoryBook.Books.Count()
                },
                CurrentCategory = null
            };

            return View(model);
        }

        public ActionResult Categories(int page = 1)
        {

```

```

ViewBag.SelectedAdminTab = "Categories";
CategoriesListViewModel model = new CategoriesListViewModel
{
    Categories = repositoryCategory.Categories
        .OrderBy(p => p.CategoryID)
        .Skip((page - 1) * PageSize)
        .Take(PageSize),
    PagingInfo = new PagingInfo
    {
        CurrentPage = page,
        ItemsPerPage = PageSize,
        TotalItems = repositoryCategory.Categories.Count()
    }
};

return View(model);
}

public ActionResult Users(int page = 1)
{
    ViewBag.SelectedAdminTab = "Users";
    UsersListViewModel model = new UsersListViewModel
    {
        Users = repositoryUser.Users
            .OrderBy(p => p.UserID)
            .Skip((page - 1) * PageSize)
            .Take(PageSize),
        PagingInfo = new PagingInfo
        {
            CurrentPage = page,
            ItemsPerPage = PageSize,
            TotalItems = repositoryUser.Users.Count()
        }
    };

    return View(model);
}

public ActionResult EditBook(int BookId)
{
    Book product = repositoryBook.Books.FirstOrDefault(p => p.BookID == BookId);
    ViewBag.SelectedAdminTab = "Books";
    return View(product);
}

public ActionResult Orders(int UserID)
{
    User user = repositoryUser.Users.FirstOrDefault(p => p.UserID == UserID);
    ViewBag.SelectedAdminTab = "Users";
    ViewBag.UserName = user.Name;
    return View(user.Orders);
}

[HttpPost]
public ActionResult EditBook(Book book, HttpPostedFileBase image, Category category)
{
    if (ModelState.IsValid)
    {
        if (image != null)
        {
            book.ImageMimeType = image.ContentType;
            book.ImageData = new byte[image.ContentLength];
            image.InputStream.Read(book.ImageData, 0, image.ContentLength);
        }
        repositoryBook.SaveBook(book);
        TempData["message"] = string.Format("Книга {0} была сохранена", book.Name);
        ViewBag.SelectedAdminTab = "Books";
        return RedirectToAction("Books");
    }
    else
    {
        ViewBag.SelectedAdminTab = "Books";
        ViewBag.ErrorValidation = true;
        return View(book);
    }
}

public ActionResult CreateBook()

```

```

    {
        ViewBag.SelectedAdminTab = "Books";
        return View("EditBook", new Book());
    }

[HttpPost]
public ActionResult DeleteBook(int BookID)
{
    Book deletedBook = repositoryBook.DeleteBook(BookID);
    if (deletedBook != null)
    {
        TempData["message"] = string.Format("Книга {0} была удалена", deletedBook.Name);
    }
    ViewBag.SelectedAdminTab = "Books";
    return RedirectToAction("Books");
}

public PartialViewResult DropdownCategories(string category)
{
    IEnumerable<Category> categories = repositoryCategory.Categories
        .Select(x => x)
        .Distinct()
        .OrderBy(x => x.Name);

    ViewBag.SelectedCategory = category;
    return PartialView(categories);
}

public ViewResult CreateCategory()
{
    ViewBag.SelectedAdminTab = "Categories";
    return View("EditCategory", new Category());
}

public ViewResult EditCategory(int CategoryID)
{
    Category category = repositoryCategory.Categories.FirstOrDefault(p => p.CategoryID == CategoryID);
    ViewBag.SelectedAdminTab = "Categories";
    return View(category);
}

[HttpPost]
public ActionResult DeleteCategory(int CategoryID)
{
    Category deletedCategory = repositoryCategory.DeleteCategory(CategoryID);
    if (deletedCategory != null)
    {
        TempData["message"] = string.Format("Категория {0} была удалена", deletedCategory.Name);
    }
    ViewBag.SelectedAdminTab = "Categories";
    return RedirectToAction("Categories");
}

[HttpPost]
public ActionResult EditCategory(Category category)
{
    if (ModelState.IsValid)
    {
        repositoryCategory.SaveCategory(category);
        TempData["message"] = string.Format("Категория {0} была сохранена", category.Name);
        ViewBag.SelectedAdminTab = "Categories";
        return RedirectToAction("Categories");
    }
    else
    {
        ViewBag.SelectedAdminTab = "Categories";
        ViewBag.ErrorValidation = true;
        return View(category);
    }
}

public ViewResult EditUser(int UserID)
{
    User user = repositoryUser.Users.FirstOrDefault(p => p.UserID == UserID);
    ViewBag.SelectedAdminTab = "Users";
    return View(user);
}

```

```

[HttpPost]
public ActionResult EditUser(User user)
{
    if (ModelState.IsValid)
    {
        repositoryUser.SaveUser(user);
        TempData["message"] = string.Format("Пользователь {0} был сохранен", user.Name);
        ViewBag.SelectedAdminTab = "Users";
        return RedirectToAction("Users");
    }
    else
    {
        ViewBag.SelectedAdminTab = "Users";
        ViewBag.ErrorValidation = true;
        return View(user);
    }
}

[HttpPost]
public ActionResult DeleteUser(int UserID)
{
    User deletedUser = repositoryUser.DeleteUser(UserID);
    if (deletedUser != null)
    {
        TempData["message"] = string.Format("Пользователь {0} был удален", deletedUser.Name);
    }
    ViewBag.SelectedAdminTab = "Users";
    return RedirectToAction("Users");
}

public ViewResult CreateUser()
{
    ViewBag.SelectedAdminTab = "Users";
    return View("EditUser", new User());
}

}
}

```

BookController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using BookStore.WebUI.Models;
using System.Configuration;

namespace BookStore.WebUI.Controllers
{
    public class BookController : Controller
    {
        private IBookRepository repository;
        public int PageSize = Convert.ToInt32(ConfigurationManager.AppSettings["NumberOfBooksOnThePage"]); //9

        public BookController(IBookRepository bookRepository)
        {
            this.repository = bookRepository;
        }

        public ViewResult List(Category category, int page = 1)
        {
            ViewBag.SelectedCategory = category;
            bool ShowAllBooks = category == null;
            int CategoryID = category != null ? category.CategoryID : 1 ;

            BooksListViewModel model = new BooksListViewModel
            {
                Books = repository.Books
                    .Where(p => ShowAllBooks || p.Category.CategoryID == CategoryID)
                    .OrderBy(p => p.BookID)
                    .Skip((page - 1) * PageSize)
                    .Take(PageSize),
            }
        }
    }
}

```

```

        PagingInfo = new PagingInfo
        {
            CurrentPage = page,
            ItemsPerPage = PageSize,
            TotalItems = (category == null || category.CategoryID == -1)?
                repository.Books.Count() :
                category.Books.Count()
        },
        CurrentCategory = category
    };
    return View(model);
}

public FileContentResult GetImage(int BookId)
{
    Book prod = repository.Books.FirstOrDefault(p => p.BookID == BookId);
    if (prod != null)
    {
        return File(prod.ImageData, prod.ImageMimeType);
    }
    else
    {
        return null;
    }
}

public ViewResult BookDetails(int BookId)
{
    Book book = repository.Books.FirstOrDefault(p => p.BookID == BookId);
    ViewBag.SelectedCategory = book.Category;
    return View(book);
}

public ViewResult SearchBooks(string NameBookSearch, int page = 1)
{
    IEnumerable<Book> FindedBooks = repository.Books.Where(p => p.Name.Contains(NameBookSearch));
    BooksListViewModel model = new BooksListViewModel
    {
        Books = FindedBooks
            .OrderBy(p => p.BookID)
            .Skip((page - 1) * PageSize)
            .Take(PageSize),
        PagingInfo = new PagingInfo
        {
            CurrentPage = page,
            ItemsPerPage = PageSize,
            TotalItems = FindedBooks.Count()
        },
        CurrentCategory = null
    };

    ViewBag.NameBookSearch = NameBookSearch;
    return View(model);
}

[HttpPost]
public ActionResult SearchBooks(string NameBookSearch)
{
    int page = 1;
    IEnumerable<Book> FindedBooks = repository.Books.Where(p => p.Name.Contains(NameBookSearch));
    BooksListViewModel model = new BooksListViewModel
    {
        Books = FindedBooks
            .OrderBy(p => p.BookID)
            .Skip((page - 1) * PageSize)
            .Take(PageSize),
        PagingInfo = new PagingInfo
        {
            CurrentPage = page,
            ItemsPerPage = PageSize,
            TotalItems = FindedBooks.Count()
        },
        CurrentCategory = null
    };

    ViewBag.NameBookSearch = NameBookSearch;
    return View("SearchBooks", model);
}

```

```

    }
}
}

```

CartController.cs

```

using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using BookStore.WebUI.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.WebUI.Controllers
{
    public class CartController : Controller
    {
        private IBookRepository repository;
        private IOrderProcessor orderProcessor;
        public CartController(IBookRepository repo, IOrderProcessor proc)
        {
            repository = repo;
            orderProcessor = proc;
        }

        public ActionResult Index(Cart cart, string returnUrl)
        {
            return View(new CartIndexViewModel { Cart = cart, ReturnUrl = returnUrl });
        }

        public RedirectToRouteResult AddToCart(Cart cart, int BookID, string returnUrl)
        {
            Book book = repository.Books.FirstOrDefault(p => p.BookID == BookID);
            if (book != null)
            {
                cart.AddItem(book, 1);
            }
            return RedirectToAction("Index", new { returnUrl });
        }

        public RedirectToRouteResult RemoveFromCart(Cart cart, int BookID, string returnUrl)
        {
            Book book = repository.Books.FirstOrDefault(p => p.BookID == BookID);
            if (book != null)
            {
                cart.RemoveLine(book);
            }
            return RedirectToAction("Index", new { returnUrl });
        }

        public PartialViewResult Summary(Cart cart)
        {
            return PartialView(cart);
        }

        [HttpPost]
        public ActionResult Checkout(Cart cart, ShippingDetails shippingDetails)
        {
            ViewBag.ServerValidation = true;
            if (cart.Lines.Count() == 0)
            {
                ModelState.AddModelError("", "Извините, ваша корзина пуста!");
            }
            if (ModelState.IsValid)
            {
                orderProcessor.ProcessOrder(cart, shippingDetails, (User)Session["CurrentUser"]);
                cart.Clear();
                ViewBag.ServerValidation = false;
                return View("Completed");
            }
            else
            {
                ViewBag.ErrorValidation = true;
                return View(shippingDetails);
            }
        }
    }
}

```

```

    }
}

public ActionResult Checkout()
{
    ViewBag.ServerValidation = true;
    return View(new ShippingDetails());
}
}
}

```

CategoryController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Controllers
{
    public class CategoryController : Controller
    {
        private ICategoryRepository repository;
        public CategoryController(ICategoryRepository repo)
        {
            repository = repo;
        }

        public PartialViewResult Menu(Category category = null)
        {
            ViewBag.SelectedCategoryName = (category == null ? "" : category.Name);
            IEnumerable<Category> categories = repository.Categories
                .Select(x => x)
                .Distinct()
                .OrderBy(x => x.Name);
            return PartialView(categories);
        }
    }
}

```

ContactsController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.WebUI.Controllers
{
    public class ContactsController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.ActiveTab = "Contacts";
            return View();
        }
    }
}

```

HelpController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.WebUI.Controllers
{

```

```

public class HelpController : Controller
{
    public ActionResult Index()
    {
        ViewBag.ActiveTab = "Help";
        return View();
    }
}

```

PaymentController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace BookStore.WebUI.Controllers
{
    public class PaymentController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.ActiveTab = "Payment";
            return View();
        }
    }
}

```

UserController.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;
using BookStore.WebUI.Infrastructure.Abstract;

namespace BookStore.WebUI.Controllers
{
    public class UserController : Controller
    {
        private IUserRepository repository;

        public UserController(IUserRepository repo)
        {
            repository = repo;
        }

        [ChildActionOnly]
        public PartialViewResult Registration(User user = null)
        {
            return PartialView(user == null ? new User() : user);
        }

        [HttpPost]
        public ActionResult RegistrationPost(User user)
        {
            if (ModelState.IsValid)
            {
                repository.SaveUser(user);
                return RedirectToAction("List", "Book");
            }
            else
            {
                return View(user);
            }
        }

        [ChildActionOnly]
        public PartialViewResult Authorization()
        {
            return PartialView();
        }
    }
}

```



```

[HttpPost]
public ActionResult AuthorizationPost(string Login, string Password)
{
    Session["CurrentUser"] = null;
    User user = repository.GetByName(Login);
    if (user != null && user.Password == Password)
    {
        Session["CurrentUser"] = user;
    }

    return RedirectToAction("List", "Book");
}

[ChildActionOnly]
public PartialViewResult NavMenu(string ActiveTab)
{
    ViewBag.ActiveTab = ActiveTab;
    return PartialView();
}
}
}

```

PagingHelpers.cs

```

using System;
using System.Text;
using System.Web.Mvc;
using BookStore.WebUI.Models;

namespace BookStore.WebUI.HtmlHelpers
{
    public static class PagingHelpers
    {
        public static MvcHtmlString PageLinks( this HtmlHelper html,
                                                PagingInfo pagingInfo,
                                                Func<int, string> pageUrl)
        {
            StringBuilder result = new StringBuilder();
            TagBuilder Nav = new TagBuilder("nav");

            //Первая страница
            TagBuilder ul = new TagBuilder("ul");
            ul.AddCssClass("pagination");
            TagBuilder liStart = new TagBuilder("li");
            TagBuilder aStart = new TagBuilder("a");
            aStart.MergeAttribute("href", pageUrl(1));
            TagBuilder span = new TagBuilder("span");
            span.MergeAttribute("aria-hidden", "true");
            span.InnerHtml += "&laquo;";
            aStart.InnerHtml += span.ToString();
            liStart.InnerHtml += aStart.ToString();
            ul.InnerHtml += liStart.ToString();

            //все остальные страницы
            for (int i = 1; i <= pagingInfo.TotalPages; i++)
            {
                TagBuilder li = new TagBuilder("li");
                if (i == pagingInfo.CurrentPage)
                    li.AddCssClass("active");

                TagBuilder a = new TagBuilder("a");
                a.MergeAttribute("href", pageUrl(i));
                a.InnerHtml = i.ToString();

                li.InnerHtml += a.ToString();
                ul.InnerHtml += li.ToString();
            }

            //последняя страница
            TagBuilder liEnd = new TagBuilder("li");
            TagBuilder aEnd = new TagBuilder("a");
            aEnd.MergeAttribute("href", pageUrl(pagingInfo.TotalPages));
            TagBuilder spanEnd = new TagBuilder("span");
            spanEnd.MergeAttribute("aria-hidden", "true");

```

```

        spanEnd.InnerHtml += ">";
        aEnd.InnerHtml += spanEnd.ToString();
        liEnd.InnerHtml += aEnd.ToString();
        ul.InnerHtml += liEnd.ToString();

        Nav.InnerHtml += ul.ToString();
        result.Append(Nav.ToString());

        return MvcHtmlString.Create(result.ToString());
    }
}

```

CategoryModelBinderProvider.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using BookStore.Domain.Abstract;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Infrastructure.Concrete
{
    public class CategoryModelBinderProvider : IModelBinderProvider
    {
        public IModelBinder GetBinder(Type modelType)
        {
            if (typeof(Category) != modelType)
                return null;
            return new CategoryModelBinder();
        }
    }

    public class CategoryModelBinder : IModelBinder
    {
        public object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
        {
            Category category = null;

            //Сначала попробуем найти название категории из данных маршрута

            ValueProviderResult value = bindingContext.ValueProvider.GetValue("category");
            ICategoryRepository repository = DependencyResolver.Current.GetService<ICategoryRepository>();

            if (value != null)
            {
                if (string.IsNullOrEmpty(value.AttemptedValue))
                    return null;

                string NameCategory = value.AttemptedValue;
                category = repository.GetByName(NameCategory);
            }

            //Попробуем получить категорию из данных формы
            else
            {
                ValueProviderResult valueName = bindingContext.ValueProvider.GetValue("Name");
                ValueProviderResult valueDescription = bindingContext.ValueProvider.GetValue("Description");

                if (valueName == null)
                    return null;

                string Name = valueName.AttemptedValue;
                string Description = valueDescription.AttemptedValue;

                category = repository.GetByName(Name);
                if (category == null)
                    category = new Category();

                category.Name = Name;
                category.Description = Description;
            }

            return category;
        }
    }
}

```

}

FormsAuthProvider.cs

```
using System.Web.Security;
using BookStore.WebUI.Infrastructure.Abstract;

namespace BookStore.WebUI.Infrastructure.Concrete
{
    public class FormsAuthProvider : IAuthProvider
    {
        public bool Authenticate(string username, string password)
        {
            bool result = FormsAuthentication.Authenticate(username, password);
            if (result)
            {
                FormsAuthentication.SetAuthCookie(username, false);
            }
            return result;
        }
    }
}
```

NinjectDependencyResolver.cs

```
using System;
using System.Web.Mvc;
using System.Web.Routing;
using Ninject;
using Moq;
using BookStore.Domain.Entities;
using BookStore.Domain.Abstract;
using BookStore.Domain.Concrete;
using System.Collections.Generic;
using System.Linq;
using System.Configuration;
using BookStore.WebUI.Infrastructure.Abstract;
using BookStore.WebUI.Infrastructure.Concrete;

namespace BookStore.WebUI.Infrastructure
{
    public class NinjectDependencyResolver : IDependencyResolver
    {
        private IKernel kernel;
        public NinjectDependencyResolver()
        {
            kernel = new StandardKernel();
            AddBindings();
        }
        public object GetService(Type serviceType)
        {
            return kernel.TryGet(serviceType);
        }
        public IEnumerable<object> GetServices(Type serviceType)
        {
            return kernel.GetAll(serviceType);
        }
        private void AddBindings()
        {
            kernel.Bind<IBookRepository>().To<EFBookRepository>();
            kernel.Bind<ICategoryRepository>().To<EFCategoryRepository>();
            kernel.Bind<IUserRepository>().To<EFUserRepository>();
            kernel.Bind<IOrderRepository>().To<EFOrderRepository>();
            kernel.Bind<IEFDbContext>().To<EFDbContext>();

            EmailSettings emailSettings = new EmailSettings
            {
                MailToAddress = ConfigurationManager.AppSettings["LogistMail"],
                MailFromAddress = ConfigurationManager.AppSettings["AdminMail"],
                UseSsl = true,
                Username = ConfigurationManager.AppSettings["AdminMail"],
                Password = ConfigurationManager.AppSettings["AdminPasswordMail"],
                ServerName = ConfigurationManager.AppSettings["AdminServerMail"],
                ServerPort = Convert.ToInt32(ConfigurationManager.AppSettings["AdminMailPort"]),
                FileLocation = ConfigurationManager.AppSettings["MailsLocation"],
                WriteAsFile = bool.Parse(ConfigurationManager.AppSettings["Email.WriteAsFile"] ?? "false")
            };
            kernel.Bind<IOrderProcessor>().To<EmailOrderProcessor>().WithConstructorArgument("settings", emailSettings);
            kernel.Bind<IAuthProvider>().To<FormsAuthProvider>();
        }
    }
}
```

```

    }
}

}

CartIndexViewModel.cs

using BookStore.Domain.Entities;

namespace BookStore.WebUI.Models
{
    public class CartIndexViewModel
    {
        public Cart Cart { get; set; }
        public string returnUrl { get; set; }
    }
}

CategoriesListViewModel.cs

using System.Collections.Generic;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Models
{
    public class CategoriesListViewModel
    {
        public IEnumerable<Category> Categories { get; set; }
        public PagingInfo PagingInfo { get; set; }
    }
}

LoginViewModel.cs

using System.ComponentModel.DataAnnotations;

namespace BookStore.WebUI.Models
{
    public class LoginViewModel
    {
        [Required]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
}

PagingInfo.cs

using System;

namespace BookStore.WebUI.Models
{
    public class PagingInfo
    {
        public int TotalItems { get; set; }
        public int ItemsPerPage { get; set; }
        public int CurrentPage { get; set; }
        public int TotalPages
        {
            get { return (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage); }
        }
    }
}

ProductsListViewModel.cs

using System.Collections.Generic;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Models
{
    public class BooksListViewModel
    {
        public IEnumerable<Book> Books { get; set; }
        public PagingInfo PagingInfo { get; set; }
        public Category CurrentCategory { get; set; }
    }
}

UsersListViewModel.cs

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using BookStore.Domain.Entities;

namespace BookStore.WebUI.Models
{
    public class UsersListViewModel
    {
        public IEnumerable<User> Users { get; set; }
        public PagingInfo PagingInfo { get; set; }
    }
}

```

Index.cshtml

```

@{
    ViewBag.Title = "О магазине";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="page-header">
    <h2>Книжный магазин "HELLO WORLD" </h2>
</div>
<p>
    
    Интернет магазин "HELLO WORLD" предлагает книги, учебники по программированию и многое другое.
    <br />
    <br />
    Покупать товары через интернет очень удобно, так как Вы можете ознакомиться с представленными книгами в интернет-магазине, а потом заказать или купить книги, которые вам понравились. На сегодняшний день книжный магазин предоставляет покупателям доставку книг Укрпочтой, перевозчиком «Новая почта» и самовывоз.
    <br />
    <br />
    Книжный магазин "HELLO WORLD" - это всегда гибкая ценовая политика. Например, мы создаем особые условия для постоянных покупателей и для того, чтобы Вы, наш читатель, захотели приобрести книги именно в нашем книжном магазине. Специально для покупателей книжного магазина разработана программа лояльности, участвуя в которой можно покупать с постоянно увеличивающейся скидкой. Уже сегодня вы можете сделать покупку в нашем книжном магазине и не платить за ее доставку, скачать флаер и купить книгу со скидкой и это только начало...
    <br />
    <br />
    Ассортимент книжного магазина формируется опытными товароведами, которые заказывают популярные и редкие книги, умело подбирают книги в рубрики новинки и распродажа.
    </p>

```

Login.cshtml

```

@model BookStore.WebUI.Models.LoginViewModel
@{
    ViewBag.Title = "Admin: Log In";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

    @using (Html.BeginForm("Login", "Account", FormMethod.Post, new { @class = "form-horizontal" }))
    {
        <div class="page-header">
            <h2>Авторизация</h2>
            <h3>Пожалуйста введите логи и пароль администратора</h3>
        </div>

        if (ViewBag.ErrorValidation != null && ViewBag.ErrorValidation == true)
        {
            <div class="alert alert-danger" role="alert">
                <strong>ОШИБКИ:</strong>
                <strong>@Html.ValidationSummary()</strong>
            </div>
        }

        string classForUserName = (ViewContext.ViewData.ModelState.IsValidField("UserName") ? "" : "has-error");
        string classForPassword = (ViewContext.ViewData.ModelState.IsValidField("Password") ? "" : "has-error");
    }

```

```

<div class="form-group" @classForUserName">
  <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.UserName)</label>
  <div class="col-sm-9">
    @Html.TextBoxFor(x => x.UserName, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
  </div>
</div>

<div class="form-group" @classForPassword">
  <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Password)</label>
  <div class="col-sm-9">
    @Html.TextBoxFor(x => x.Password, new { @class = "form-control", type = "password", id = "formGroupInputLarge" })
  </div>
</div>

<div class="form-group">
  <div class="col-sm-offset-3 col-sm-9">
    <input type="submit" value="Бойти" class="btn btn-success" />
  </div>
</div>
}
</div>

```

Books.cshtml

```

@model BookStore.WebUI.Models.BooksListViewModel

@{
    ViewBag.Title = "Книги";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

    <table class="table table-hover">
        <tr>
            <th>
                Название
            </th>
            <th>
                Автор
            </th>
            <th>
                Издатель
            </th>
            <th>
                Год издания
            </th>
            <th>
                ISBN
            </th>
            <th>
                Кол-во страниц
            </th>
            <th>
                Цена
            </th>
            <th colspan="2">
                @Html.ActionLink("Добавить новую книгу", "CreateBook", "Admin", null, new { @class = "btn btn-primary", type = "button" })
            </th>
        </tr>

        @foreach (var item in Model.Books)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Author)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Publisher)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Year)
                </td>
            </tr>
        }
    </table>

```

```

        <td>
            @Html.DisplayFor(modelItem => item.ISBN)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.NumberOfPages)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Price)
        </td>
        <td>
            @Html.ActionLink("Редактировать", "EditBook", "Admin", new { BookID = item.BookID }, new { id = item.BookID, @class
= "btn btn-success", type = "button" })
        </td>
        <td>
            @using (Html.BeginForm("DeleteBook", "Admin"))
            {
                @Html.Hidden("BookID", item.BookID)
                <input type="submit" value="Удалить" class="btn btn-danger" />
            }
        </td>
    </tr>
</table>

</div>

<br />
<div class="row" id="numerator">
<div class="col-xs-6 col-lg-4"></div>
<div class="btn-group col-xs-6 col-lg-4" role="group" aria-label="First group">
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("Books", "Admin", new { page = x }))
</div>
</div><!--/row-->

```

Categories.cshtml

```

@model BookStore.WebUI.Models.CategoriesListViewModel

@{
    ViewBag.Title = "Категории";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

    <table class="table table-hover">
        <tr>
            <th>
                Название
            </th>
            <th>
                Описание
            </th>
            <th colspan="2">
                @Html.ActionLink("Добавить новую категорию", "CreateCategory", "Admin", null, new { @class = "btn btn-primary", type
= "button" })
            </th>
        </tr>

        @foreach (var item in Model.Categories)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.Name)
                </td>
                <td>
                    @Html.DisplayFor(modelItem => item.Description)
                </td>
                <td>
                    @Html.ActionLink("Редактировать", "EditCategory", "Admin", new { CategoryID=item.CategoryID }, new { id =
item.CategoryID, @class = "btn btn-success", type = "button" })
                </td>
                <td>
                    @using (Html.BeginForm("DeleteCategory", "Admin"))
                    {
                        @Html.Hidden("CategoryID", item.CategoryID)
                        <input type="submit" value="Удалить" class="btn btn-danger" />
                    }
                </td>
            </tr>
        }
    </table>
</div>

```

```

    }
  </td>
</tr>
}
</table>

</div>

<br />
<div class="row" id="numerator">
  <div class="col-xs-6 col-lg-4"></div>
  <div class="btn-group col-xs-6 col-lg-4" role="group" aria-label="First group">
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("Categories", "Admin", new { page = x }))
  </div>
</div><!--/row-->

```

DropdownCategories.cshtml

```

@model IEnumerable<BookStore.Domain.Entities.Category>

<select name="category" class="form-control" id="formGroupInputLarge">
  <option>
    @ViewBag.SelectedCategory
  </option>
  @foreach (var item in Model)
  {
    if (item.Name != @ViewBag.SelectedCategory)
    {
      <option>
        @item.Name
      </option>
    }
  }
</select>

```

EditBook.cshtml

```

@model BookStore.Domain.Entities.Book
@{
  ViewBag.Title = "Редактирование" + @Model.Name;
  Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

  @using (Html.BeginForm("EditBook", "Admin", FormMethod.Post, new { enctype = "multipart/form-data", @class = "form-horizontal"
  )))
  {
    @Html.Hidden("BookID", @Model.BookID)
    <br />
    <blockquote>
      <p>Редактирование книги: @Model.Name</p>
      <footer>ID <cite title="Source Title">@Model.BookID</cite></footer>
    </blockquote>

    if (ViewBag.ErrorValidation != null && ViewBag.ErrorValidation == true)
    {
      <div class="alert alert-danger" role="alert">
        <strong>ОШИБКИ:</strong>
        <strong>@Html.ValidationSummary()</strong>
      </div>
    }

    string classForName = (ViewContext.ViewData.ModelState.IsValidField("Name") ? "" : "has-error");
    string classForCategory = (ViewContext.ViewData.ModelState.IsValidField("Category") ? "" : "has-error");
    string classForAuthor = (ViewContext.ViewData.ModelState.IsValidField("Author") ? "" : "has-error");
    string classForPublisher = (ViewContext.ViewData.ModelState.IsValidField("Publisher") ? "" : "has-error");
    string classForYear = (ViewContext.ViewData.ModelState.IsValidField("Year") ? "" : "has-error");
    string classForISBN = (ViewContext.ViewData.ModelState.IsValidField("ISBN") ? "" : "has-error");
    string classForNumberOfPages = (ViewContext.ViewData.ModelState.IsValidField("NumberOfPages") ? "" : "has-error");
    string classForPrice = (ViewContext.ViewData.ModelState.IsValidField("Price") ? "" : "has-error");
    string classForDescription = (ViewContext.ViewData.ModelState.IsValidField("Description") ? "" : "has-error");

    <div class="form-group @classForName">

```



```

<label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Name)</label>
<div class="col-sm-9">
    @Html.TextBoxFor(x => x.Name, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
</div>
</div>

<div class="form-group" @classForCategory">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Category)</label>
    <div class="col-sm-9">
        @ { string categoryName = (@Model.Category == null ? "" : @Model.Category.Name); }
        @ { Html.RenderAction("DropdownCategories", "Admin", new { category = @categoryName }); }
    </div>
</div>

<div class="form-group" @classForAuthor">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Author)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.Author, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForPublisher">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Publisher)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.Publisher, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForYear">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Year)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.Year, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForISBN">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.ISBN)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.ISBN, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForNumberOfPages">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model =>
model.NumberOfPages)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.NumberOfPages, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForPrice">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Price)</label>
    <div class="col-sm-9">
        @Html.TextBoxFor(x => x.Price, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
    </div>
</div>

<div class="form-group" @classForDescription">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Description)</label>
    <div class="col-sm-9">
        @Html.TextAreaFor(x => x.Description, new { @class = "form-control", @rows = 5, type = "text", id = "formGroupInputLarge"
    </div>
</div>
})

<div class="form-group">
    <label class="col-sm-3 control-label" for="formGroupInputLarge">Картинка</label>
    <div class="col-sm-9">
        @if (Model.ImageData == null)
        {
            @:Нет картинки
        }
        else
        {
            
        }
    </div>
</div>

```

```

        <input type="file" name="Image" class="btn btn-default" />
    </div>
</div>

<div class="form-group">
    <div class="col-sm-offset-3 col-sm-9">
        <input type="submit" value="Сохранить" class="btn btn-success" />
        @Html.ActionLink("Отменить и вернуться к списку", "Books", "Admin", null, new { @class = "btn btn-default", type =
"button" })
    </div>
</div>

}

</div>

```

EditCategory.cshtml

@model BookStore.Domain.Entities.Category

```

@{
    ViewBag.Title = "Редактирование категории";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

```

```

<div class="row">

    @using (Html.BeginForm("EditCategory", "Admin", FormMethod.Post, new { @class = "form-horizontal" }))
    {
        <br />
        <blockquote>
            <p>Редактирование категории: @Model.Name</p>
            <footer>ID <cite title="Source Title">@Model.CategoryID</cite></footer>
        </blockquote>

        @Html.Hidden("CategoryID", @Model.CategoryID)

        if (ViewBag.ErrorValidation != null && ViewBag.ErrorValidation == true)
        {
            <div class="alert alert-danger" role="alert">
                <strong>ОШИБКИ:</strong>
                <strong>@Html.ValidationSummary()</strong>
            </div>
        }

        string classForName = (ViewContext.ViewData.ModelState.IsValidField("Name") ? "" : "has-error");
        string classForDescription = (ViewContext.ViewData.ModelState.IsValidField("Description") ? "" : "has-error");

        <div class="form-group" @classForName>
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Name)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Name, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group" @classForDescription>
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Description)</label>
            <div class="col-sm-9">
                @Html.TextAreaFor(x => x.Description, new { @class = "form-control", @rows = 5, type = "text", id = "formGroupInputLarge"
            </div>
        </div>

        <div class="form-group">
            <div class="col-sm-offset-3 col-sm-9">
                <input type="submit" value="Сохранить" class="btn btn-success" />
                @Html.ActionLink("Отменить и вернуться к списку", "Categories", "Admin", null, new { @class = "btn btn-default", type =
"button" })
            </div>
        </div>
    }

</div>

```

EditUser.cshtml

```
@model BookStore.Domain.Entities.User

@{
    ViewBag.Title = "Редактирование пользователя";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

    @using (Html.BeginForm("EditUser", "Admin", FormMethod.Post, new { @class = "form-horizontal" }))
    {
        <br />

        @Html.Hidden("UserID", @Model.UserID)
        if (ViewBag.ErrorValidation != null && ViewBag.ErrorValidation == true)
        {
            <div class="alert alert-danger" role="alert">
                <strong>ОШИБКИ:</strong>
                <strong>@Html.ValidationSummary()</strong>
            </div>
        }

        string classForName = (ViewContext.ViewData.ModelState.IsValidField("Name") ? "" : "has-error");
        string classForEmail = (ViewContext.ViewData.ModelState.IsValidField("Email") ? "" : "has-error");
        string classForPassword = (ViewContext.ViewData.ModelState.IsValidField("Password") ? "" : "has-error");

        <div class="form-group" @classForName">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Name)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Name, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group" @classForEmail">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Email)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Email, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group" @classForPassword">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Password)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Password, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-sm-offset-3 col-sm-9">
                <input type="submit" value="Сохранить" class="btn btn-success" />
                @Html.ActionLink("Отменить и вернуться к списку", "Users", "Admin", null, new { @class = "btn btn-default", type = "button" })
            </div>
        </div>
    }
</div>
```

Orders.cshtml

```
@model IEnumerable<BookStore.Domain.Entities.Order>

@{
    ViewBag.Title = "Редактирование категории";
    Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

    <div class="page-header">
        <h2>Заказы покупателя @ViewBag.UserName</h2>
    </div>
```

```

<table class="table table-hover">
<thead>
  <tr>
    <th>Дата</th>
    <th>Книга</th>
    <th>Количество</th>
    <th>Цена</th>
    <th>Сумма</th>
  </tr>
</thead>
<tbody>

  @{ decimal totalSum = 0; decimal sum = 0; }
  @foreach (BookStore.Domain.Entities.Order order in Model)
  {
    foreach(BookStore.Domain.Entities.OrderLine line in order.OrderLines)
    {

      sum = line.Quantity * line.Price;
      totalSum += (line.Quantity * line.Price);

      <tr>
        <th>
          @Html.DisplayFor(x => order.Date)
        </th>
        <th>
          @Html.DisplayFor(x => line.Book.Name)
        </th>
        <th>
          @Html.DisplayFor(x => line.Quantity)
        </th>
        <th>
          @Html.DisplayFor(x => line.Price)
        </th>
        <th>
          @sum
        </th>
      </tr>
    }
  }

  <tr>
    <th colspan="4" class="text-right">
      Всего:
    </th>
    <th>
      @totalSum
    </th>
  </tr>

</tbody>
</table>

</div>

```

Users.cshtml

```

@model BookStore.WebUI.Models.UsersListViewModel

@{
  ViewBag.Title = "Пользователи";
  Layout = "~/Views/Shared/_AdminLayout.cshtml";
}

<div class="row">

  <table class=" table table-hover">
    <tr>
      <th>
        Логин
      </th>
      <th>
        Email
      </th>

```

```

        <th colspan="2" class="text-center">
            @Html.ActionLink("Добавить нового пользователя", "CreateUser", "Admin", null, new { @class = "btn btn-primary", type =
"button" })
        </th>

    </tr>

    @foreach (var item in Model.Users)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td class="text-right">
                @Html.ActionLink("Заказы", "Orders", "Admin", new { UserID = item.UserID }, new { id = item.UserID, @class = "btn btn-
success", type = "button" })
                @Html.ActionLink("Редактировать", "EditUser", "Admin", new { UserID = item.UserID }, new { id = item.UserID, @class =
"btn btn-success", type = "button" })
            </td>
            <td>
                @using (Html.BeginForm("DeleteUser", "Admin"))
                {
                    @Html.Hidden("UserID", item.UserID)
                    <input type="submit" value="Удалить" class="btn btn-danger" />
                }
            </td>
        </tr>
    }

</table>

</div>

<br />
<div class="row" id="numerator">
    <div class="col-xs-6 col-lg-4"></div>
    <div class="btn-group col-xs-6 col-lg-4" role="group" aria-label="First group">
        @Html.PageLinks(Model.PagingInfo, x => Url.Action("Users", "Admin", new { page = x}))
    </div>
</div><!--/row-->

```

Book.cshtml

```

@model BookStore.Domain.Entities.Book

@using (Html.BeginForm("AddToCart", "Cart"))
{
    @Html.HiddenFor(x => x.BookID)
    @Html.Hidden("returnUrl", Request.Url.PathAndQuery)

    <div class="col-xs-6 col-lg-4">
        <div class="thumbnail">

            @if (Model.ImageData != null)
            {
                
            }
            else
            {
                
            }

            <div class="caption">
                <h3>@Model.Name</h3>
                <p>
                    <a class="btn btn-default" href="@Url.Action("BookDetails", "Book", new { BookID = @Model.BookID })" role="button">Детали
                    &raquo;</a>

                    <button type="submit" class="btn btn-primary" role="button">
                        <span class="glyphicon glyphicon-shopping-cart" aria-hidden="true"></span> В корзину
                    </button>
                </p>
            </div>

```

```

    </div>
</div><!--/.col-xs-6.col-lg-4-->

```

```

}

```

BookDetails.cshtml

```

@model BookStore.Domain.Entities.Book

```

```

@{
    ViewBag.Title = "Книги";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

```

```

<div class="container">

    <div class="row row-offcanvas row-offcanvas-right">

        <div class="col-xs-12 col-sm-9">

            <div class="row">
                <h2>@Model.Name</h2>
            </div>

            <div class="row">

                <div class="col-xs-6 col-lg-4">
                    @if (Model.ImageData != null)
                    {
                        
                    }
                    else
                    {
                        
                    }
                </div>

                <div class="col-xs-6 col-lg-8">
                    <div class="caption">

                        @using (Html.BeginForm("AddToCart", "Cart"))
                        {
                            @Html.HiddenFor(t => t.BookID)
                            @Html.Hidden("returnUrl", Request.Url.PathAndQuery)
                            <table class="table">
                                <tr>
                                    <td>@Html.DisplayNameFor(model => model.Author)</td>
                                    <td>@Model.Author</td>
                                </tr>
                                <tr>
                                    <td>@Html.DisplayNameFor(model => model.Publisher)</td>
                                    <td>@Model.Publisher</td>
                                </tr>
                                <tr>
                                    <td>@Html.DisplayNameFor(model => model.ISBN)</td>
                                    <td>@Model.ISBN</td>
                                </tr>
                                <tr>
                                    <td>@Html.DisplayNameFor(model => model.NumberOfPages)</td>
                                    <td>@Model.NumberOfPages</td>
                                </tr>
                                <tr>
                                    <td>@Html.DisplayNameFor(model => model.Year)</td>
                                    <td>@Model.Year</td>
                                </tr>
                                <tr>
                                    <td><strong>@Html.DisplayNameFor(model => model.Price)</strong></td>
                                    <td><strong>@Model.Price</strong></td>
                                </tr>
                            </table>

                            <button type="submit" class="btn btn-primary" role="button">
                                <span class="glyphicon glyphicon-shopping-cart" aria-hidden="true"></span> В корзину
                            </button>
                        }
                    </div>
                </div>
            </div>

```

```

        </div>
    </div>

</div><!--/row-->

<div class="row">
    <br />
    <p>@Model.Description</p>
</div>

</div><!--/.col-xs-12.col-sm-9-->

<!--Группа разделов-->
@{ string SelectedCategory = @ViewBag.SelectedCategory != null ? @ViewBag.SelectedCategory.Name : ""; }
@{ Html.RenderAction("Menu", "Category", new { category = @SelectedCategory }); }

</div><!--/row-->
</div><!--/.container-->

```

List.cshtml

```

@model BookStore.WebUI.Models.BooksListViewModel

@{
    ViewBag.Title = "Книги";
}

<div class="container">

    <div class="row row-offcanvas row-offcanvas-right">

        <div class="col-xs-12 col-sm-9">
            <p class="pull-right visible-xs">
                <button type="button" class="btn btn-primary btn-xs" data-toggle="offcanvas">Toggle nav</button>
            </p>
            @if (Model.CurrentCategory != null)
            {
                <div class="jumbotron">
                    <h1>@Model.CurrentCategory.Name</h1>
                    <p>@Model.CurrentCategory.Description</p>
                </div>
            }

            <!--Контейнер книг-->
            <div>

                @{ int i = 1; bool opendiv = false; }

                @foreach (var p in Model.Books)
                {
                    if (i == 1 || ((i-1) % 3) == 0)
                    {
                        @:<div class="row">
                            opendiv = true;
                        }
                    }

                    Html.RenderPartial("Book", p);

                    if (i % 3 == 0)
                    {
                        @:</div>
                        opendiv = false;
                    }
                    i++;
                }

                @if (opendiv)
                {
                    @:</div>
                }

            <!--Нумерация-->
            <br />
            <div class="row">

```

```

<div class="col-xs-6 col-lg-4"></div>
<div class="btn-group col-xs-6 col-lg-4" role="group" aria-label="First group">
    @Html.PageLinks(Model.PagingInfo, x => Url.Action("List", "Book",
        new { page = x, category = (Model.CurrentCategory == null ? (string)null : Model.CurrentCategory.Name) }))
</div>
</div><!--/row-->

</div> <!--Контейнер книг-->

</div><!--/.col-xs-12.col-sm-9-->

<!--Группа разделов-->
@{ string SelectedCategory = @ViewBag.SelectedCategory != null ? @ViewBag.SelectedCategory.Name : ""; }
@{ Html.RenderAction("Menu", "Category", new { category = @SelectedCategory }); }

</div><!--/row-->

</div><!--/.container-->

```

SearchBooks.cshtml

```

@model BookStore.WebUI.Models.BooksListViewModel

@{
    ViewBag.Title = "Книги";
}

<div class="container">

    <div class="row row-offcanvas row-offcanvas-right">

        <div class="col-xs-12 col-sm-9">
            <p class="pull-right visible-xs">
                <button type="button" class="btn btn-primary btn-xs" data-toggle="offcanvas">Toggle nav</button>
            </p>

            <div class="page-header">
                <h2>Поиск</h2>
                <h3>Подстрока поиска <strong>@ViewBag.NameBookSearch</strong></h3>
            </div>

            <!--Контейнер книг-->
            <div>

                @{ int i = 1; bool opendiv = false; }

                @foreach (var p in Model.Books)
                {
                    if (i == 1 || ((i - 1) % 3) == 0)
                    {
                        @:<div class="row">
                            opendiv = true;
                        }

                        Html.RenderPartial("Book", p);

                        if (i % 3 == 0)
                        {
                            @:</div>
                            opendiv = false;
                        }
                        i++;
                    }

                    @if (opendiv)
                    {
                        @:</div>
                    }

                <!--Нумерация-->
                <br />
                <div class="row">
                    <div class="col-xs-6 col-lg-4"></div>
                    <div class="btn-group col-xs-6 col-lg-4" role="group" aria-label="First group">
                        @Html.PageLinks(Model.PagingInfo, x => Url.Action("SearchBooks", "Book",

```



```

new { page = x, NameBookSearch = @ViewBag.NameBookSearch })

</div>
</div><!--/row-->

<div> <!--Контейнер книг-->

</div><!--/.col-xs-12.col-sm-9-->

<!--Группа разделов-->
@{ string SelectedCategory = @ViewBag.SelectedCategory != null ? @ViewBag.SelectedCategory.Name : ""; }
@{ Html.RenderAction("Menu", "Category", new { category = @SelectedCategory }); }

</div><!--/row-->
</div><!--/.container-->

```

Checkout.cshtml

```
@model BookStore.Domain.Entities.ShippingDetails
```

```

@{
    ViewBag.Title = "Корзина";
}

<div class="row">

    @using (Html.BeginForm("Checkout", "Cart", FormMethod.Post, new { @class = "form-horizontal" }))
    {

        <div class="page-header">
            <h2>Оформление заказа</h2>
            <h3>Пожалуйста проверьте детали, и мы отправим ваши книги сегодня же!</h3>
        </div>

        if (ViewBag.ErrorValidation != null && ViewBag.ErrorValidation == true)
        {
            <div class="alert alert-danger" role="alert">
                <strong>ОШИБКИ:</strong>
                <strong>@Html.ValidationSummary()</strong>
            </div>
        }

        string classForName = (ViewContext.ViewData.ModelState.IsValidField("Name") ? "" : "has-error");

        <div class="form-group @classForName">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Name)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Name, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Address)</label>
            <div class="col-sm-9">
                @Html.TextBoxFor(x => x.Address, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group">
            <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Description)</label>
            <div class="col-sm-9">
                @Html.TextAreaFor(x => x.Description, new { @class = "form-control", @rows = 5, type = "text", id = "formGroupInputLarge" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-sm-offset-3 col-sm-9">
                <input type="submit" value="Отправить заказ" class="btn btn-success" />
            </div>
        </div>

    }

</div>

```

Completed.cshtml

```
@{
    ViewBag.Title = "Корзина";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>Спасибо!</h2>
<p>Спасибо за оформление заказа. Ваша посылка придет в ближайшее время.</p>
```

Index.cshtml

```
@model BookStore.WebUI.Models.CartIndexViewModel
```

```
@{
    ViewBag.Title = "Корзина";
}

@{
    string UserName = Session["CurrentUser"] != null ? ((BookStore.Domain.Entities.User)(Session["CurrentUser"])).Name : "Гость";
}

<h2>Ваша корзина</h2>

<table width="90%" align="center" class="table table-hover">
    <thead>
        <tr>
            <th align="center">Количество</th>
            <th align="left">Название</th>
            <th align="right">Цена</th>
            <th align="right">Сумма</th>
        </tr>
    </thead>
    <tbody>
        @foreach (var line in Model.Cart.Lines)
        {
            <tr>
                <td align="center">@line.Quantity</td>
                <td align="left">@line.Book.Name</td>
                <td align="right">@line.Book.Price.ToString("0.##") грн.</td>
                <td align="right">@((line.Quantity * line.Book.Price).ToString("0.##")) грн.</td>
            </tr>
            @using (Html.BeginForm("RemoveFromCart", "Cart"))
            {
                @Html.Hidden("BookId", line.Book.BookID)
                @Html.HiddenFor(x => x.ReturnUrl)
                <input type="submit" value="Удалить" class="btn btn-danger" />
            }
        </td>
    </tr>
    </tbody>
    <tfoot>
        <tr>
            <td colspan="3" align="right"><strong>Всего:</strong></td>
            <td align="right">
                <strong>
                    @Model.Cart.ComputeTotalValue().ToString("0.##") грн.
                </strong>
            </td>
        </tr>
    </tfoot>
</table>
<p align="center" class="actionButtons">
    <a class="btn btn-default" href="@Model.ReturnUrl">Продолжить покупки</a>

    <a class="btn btn-primary" role="button" href="@Url.Action("Checkout", "Cart")">
        <span class="glyphicon glyphicon-ok" aria-hidden="true"></span> Оформить заказ
    </a>
</p>
```

Summary.cshtml

```
@model BookStore.Domain.Entities.Cart
```

```
<a href="@Url.Action("Index", "Cart", new { returnUrl = Request.Url.PathAndQuery })">Корзина <span
class="badge">@Model.ComputeTotalValue().ToString("0.##") грн.</span></a>
```

Menu.cshtml

```
@model IEnumerable<Category>
```

```

<div class="col-xs-6 col-sm-3 sidebar-offcanvas" id="sidebar">
  <div class="list-group">

    @foreach (var link in Model)
    {
      @Html.RouteLink(link.Name,
        new
        {
          controller = "Book",
          action = "List",
          category = link.Name,
          page = 1
        },
        new
        {
          @class = (link.Name == ViewBag.SelectedCategoryName ? "list-group-item active" : "list-group-item")
        })
    }

  </div>
</div>

```

Index.cshtml

```

@{
  ViewBag.Title = "Контакты";
  Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="page-header">
  <h2>Контакты</h2>

  <h2>Интернет магазин "HELLO WORLD"</h2>
</div>

<br />
<p>
  
  <strong>Телефон:</strong> 111-11-11
  <br />
  <strong>Телефон:</strong> 222-22-22
  <br />
  <strong>Телефон:</strong> 333-33-33
  <br />
  <strong>Режим Работы:</strong> Пн-Пт (10.00-17.00)
  <br />
  <strong>Email:</strong> HelloWorld@gmail.com
  <br />
  <br />
</p>

```

_AdminLayout.cshtml

```

<!DOCTYPE html>
<html>
<head>
  <title>@ViewBag.Title</title>
  <meta charset="windows-1251">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="">
  <meta name="author" content="">
  <link rel="icon" href="~/Content/Images/Brande26x26.png">
  <link href="~/Content/bootstrap.min.css" rel="stylesheet">
  <script src="~/Scripts/jquery-1.9.1.min.js"></script>
  <script src="~/Scripts/bootstrap.min.js"></script>
  <script type="text/javascript">
    $(document).ready(function () {
      if ($(document).height() <= $(window).height())
        $("#numerator").addClass("navbar-fixed-bottom");

      $('#modal').modal();
    });
  </script>
</head>

```

```

<body>

@{
    var classBooks = "Books" == ViewBag.SelectedAdminTab ? "active" : "";
    var classCategories = "Categories" == ViewBag.SelectedAdminTab ? "active" : "";
    var classUsers = "Users" == ViewBag.SelectedAdminTab ? "active" : "";
}

<div class="container">
    <div class="row">
        <div class="col-sm-offset-1 col-sm-10">

            <br />
            <ul class="nav nav-tabs">
                <li role="presentation" class=@classBooks>
                    @Html.RouteLink("Книги", new { controller = "Admin", action = "Books" })
                </li>
                <li role="presentation" class=@classCategories>
                    @Html.RouteLink("Категории", new { controller = "Admin", action = "Categories" })
                </li>
                <li role="presentation" class=@classUsers>
                    @Html.RouteLink("Пользователи", new { controller = "Admin", action = "Users" })
                </li>
            </ul>

            <br />
            @if (TempData["message"] != null)
            {
                <div class="alert alert-info" role="alert">@TempData["message"]</div>
            }

            @RenderBody()

        </div>
    </div>
</div>

</body>

</html>

```

_Layout.cshtml

```

<!DOCTYPE html>
<html>
<head>

    <title>@ViewBag.Title</title>

    <meta charset="windows-1251">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="">
    <meta name="author" content="">
    <link rel="icon" href="~/Content/Images/Brande26x26.png">

    <link href="~/Content/bootstrap.min.css" rel="stylesheet">
    <link href="~/Content/ie10-viewport-bug-workaround.css" rel="stylesheet">
    <link href="~/Content/offcanvas.css" rel="stylesheet">
    <link href="~/Content/bootstrap.css" rel="stylesheet">

    <script src="~/Scripts/ie-emulation-modes-warning.js"></script>
    <script src="~/Scripts/jquery-1.9.1.min.js"></script>
    <script src="~/Scripts/bootstrap.js" type="text/javascript"></script>

    <!--Валидация на стороне клиента-->
    @if (ViewBag.ServerValidation == null || ViewBag.ServerValidation == false)
    {
        HtmlHelper.ClientValidationEnabled = true;
        HtmlHelper.UnobtrusiveJavaScriptEnabled = true;
        <script src="~/Scripts/jquery-1.9.1.min.js"></script>
        <script src="~/Scripts/jquery.validate.min.js"></script>
        <script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
    }

```

```

<script type="text/javascript">
    $(document).ready(function () {
        if($(document).height() <= $(window).height())
            $(".footer.footer").addClass("navbar-fixed-bottom");

        $('#modal').modal();
    });
</script>
</head>
<body>

<!-- Меню сайта -->
<nav class="navbar navbar-fixed-top navbar-inverse">
    @ { Html.RenderAction("NavMenu", "User", new { ActiveTab = ViewBag.ActiveTab }); }
</nav><!-- /.navbar -->

<!-- Модальное окно авторизации -->
<div id="modal" class="modal fade">
    @ { Html.RenderAction("Authorization", "User"); }
</div>

<!-- Модальное окно Регистрации -->
<div id="modalReg" class="modal fade">
    @ { Html.RenderAction("Registration", "User", new { user = new User() }); }
</div>

<!-- Контейнер данных -->
<div class="container">
    @RenderBody()
</div>

<footer class="footer">
    <div class="container">
        <p class="text-muted">Компьютерная академия ШАГ </p>
        <p class="text-muted">&copy; 2016 Ковалев Денис. Группа ВПУ-1311 </p>
    </div>
</footer>

<!-- Bootstrap core JavaScript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script>window.jQuery || document.write('<script src="~/Scripts/jquery-1.9.1.min.js"></script>')</script>
<script src="~/Scripts/ie10-viewport-bug-workaround.js"></script>
<script src="~/Scripts/offcanvas.js"></script>

</body>
</html>

```

Authorization.cshtml

```

@using (Html.BeginForm("AuthorizationPost", "User", FormMethod.Post))
{
    <div class="modal-dialog modal-sm">
        <div class="modal-content">
            <!-- Заголовок модального окна -->
            <div class="modal-header">
                <!--<button type="button" class="close" data-dismiss="modal" aria-hidden="true">x</button>-->
                <h4 class="modal-title">Авторизация</h4>
            </div>
            <!-- Основное содержимое модального окна -->
            <div class="modal-body">

                <input type="text" class="form-control" placeholder="Логин" aria-describedby="basic-addon1" name="Login">

                <br />

                <input type="password" class="form-control" placeholder="Пароль" aria-describedby="basic-addon1" name="Password">

            </div>
            <!-- Футер модального окна -->
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">Закрыть</button>
                <button type="submit" value="Save" class="btn btn-primary">Вход</button>
            </div>
        </div>
    </div>
}

```

```

</div>
}

NavMenu.cshtml
@{
    string classAbout = (ViewBag.ActiveTab == "About") ? "active" : " ";
    string classPayment = (ViewBag.ActiveTab == "Payment") ? "active" : " ";
    string classHelp = (ViewBag.ActiveTab == "Help") ? "active" : " ";
    string classContacts = (ViewBag.ActiveTab == "Contacts") ? "active" : " ";
}

<div class="container">
    <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Навигация</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="@Url.Action("List", "Book")"><strong>HELLO WORLD</strong></a>
    </div>

    <div id="navbar" class="collapse navbar-collapse">
        <ul class="nav navbar-nav">

            <li class="@classAbout"><a href="@Url.Action("Index", "About")">О магазине</a></li>
            <li class="@classPayment"><a href="@Url.Action("Index", "Payment")">Оплата</a></li>
            <li class="@classHelp"><a href="@Url.Action("Index", "Help")">Помощь</a></li>
            <li class="@classContacts"><a href="@Url.Action("Index", "Contacts")">Контакты</a></li>
        </ul>

        @using (Html.BeginForm("SearchBooks", "Book", FormMethod.Post, new { @class = "navbar-form navbar-left", role="search" }))
        {
            <div class="form-group">
                <input type="text" class="form-control" placeholder="введите название книги..." name="NameBookSearch"
id="NameBookSearch">
            </div>
            <button type="submit" class="btn btn-default">Поиск</button>
        }

        <ul class="nav navbar-nav navbar-right">

            @{
                string UserName = "";
                if (Session["CurrentUser"] != null)
                {
                    UserName = ((BookStore.Domain.Entities.User)(Session["CurrentUser"])).Name;
                }
            }
            @{
                if (UserName != "")
                {
                    <li>
                        <a href="#"><strong>@UserName</strong></a>
                    </li>
                    <li>
                        @Html.RenderAction("Summary", "Cart");
                    </li>
                }
                else
                {
                    <li>
                        <a href="#modalReg" data-toggle="modal" data-target="#modalReg"><strong>Регистрация</strong></a>
                    </li>
                    <li>
                        <a href="#modal" data-toggle="modal" data-target="#modal"><strong>Вход</strong></a>
                    </li>
                }
            }

        </ul>

    </div><!-- /.nav-collapse -->

```

```
</div><!-- /.container -->
```

Registration.cshtml

```
@model BookStore.Domain.Entities.User
```

```
@*<script src="~/Scripts/jquery-1.9.1.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>*@
```

```
@using (Html.BeginForm("RegistrationPost", "User", FormMethod.Post, new { @class = "form-horizontal" }))
{
```

```
    @Html.AntiForgeryToken()
```

```
    @Html.ValidationSummary(true)
```

```
    <div class="modal-dialog modal-md">
```

```
        <div class="modal-content">
```

```
            <!-- Заголовок модального окна -->
```

```
            <div class="modal-header">
```

```
                <h4 class="modal-title">Регистрация</h4>
```

```
            </div>
```

```
            <!-- Основное содержимое модального окна -->
```

```
            <div class="modal-body">
```

```
                @Html.HiddenFor(model => model.UserID)
```

```
                <div class="form-group">
```

```
                    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Name)</label>
```

```
                    <div class="col-sm-9">
```

```
                        @Html.TextBoxFor(x => x.Name, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
```

```
                        @Html.ValidationMessageFor(model => model.Name, null, new { style = "color:red" })
```

```
                    </div>
```

```
                </div>
```

```
                <div class="form-group">
```

```
                    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Password)</label>
```

```
                    <div class="col-sm-9">
```

```
                        @Html.TextBoxFor(x => x.Password, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
```

```
                        @Html.ValidationMessageFor(model => model.Password, null, new { style = "color:red" })
```

```
                    </div>
```

```
                </div>
```

```
                <div class="form-group">
```

```
                    <label class="col-sm-3 control-label" for="formGroupInputLarge">@Html.DisplayNameFor(model => model.Email)</label>
```

```
                    <div class="col-sm-9">
```

```
                        @Html.TextBoxFor(x => x.Email, new { @class = "form-control", type = "text", id = "formGroupInputLarge" })
```

```
                        @Html.ValidationMessageFor(model => model.Email, null, new { style = "color:red" })
```

```
                    </div>
```

```
                </div>
```

```
        </div>
```

```
        <!-- Футер модального окна -->
```

```
        <div class="modal-footer">
```

```
            <button type="button" class="btn btn-default" data-dismiss="modal">Закрыть</button>
```

```
            <button type="submit" value="Save" class="btn btn-primary">Регистрация</button>
```

```
        </div>
```

```
    </div>
```

```
</div>
```

```
}
```

_ViewStart.cshtml

```
@{
```

```
    Layout = "~/Views/Shared/_Layout.cshtml";
```

```
}
```

Global.asax

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Http;
```

```

using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
using BookStore.WebUI.Infrastructure;
using BookStore.WebUI.Infrastructure.Concrete;
using BookStore.Domain.Concrete;
using BookStore.Domain.Entities;
using System.Data.Entity;
using BookStore.WebUI.Binders;

namespace BookStore.WebUI
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            Database.SetInitializer<EFDbContext>(new DbInitializer());
            AreaRegistration.RegisterAllAreas();

            WebApiConfig.Register(GlobalConfiguration.Configuration);
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

            DependencyResolver.SetResolver(new NinjectDependencyResolver());
            ModelBinderProviders.BinderProviders.Add(new CategoryModelBinderProvider());
            ModelBinders.Binders.Add(typeof(Cart), new CartModelBinder());
        }
    }
}

```

Web.config

```

<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=152368
-->
<configuration>

  <configSections>
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=237468 -->
  </configSections>

  <connectionStrings>
    <add name="EFDbContext" connectionString="Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Data.mdf;Integrated
Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>

  <appSettings>
    <add key="webpages:Version" value="2.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="PreserveLoginUrl" value="true" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="Email.WriteAsFile" value="true"/>
    <add key="NumberOfBooksOnThePage" value="9"/>
    <add key="AdminMail" value="asdfjkdksdfjsdfgvjmgjtk@mail.ru"/>
    <add key="LogistMail" value="denis.grin1618@gmail.com"/>
    <add key="AdminPasswordMail" value="admin16181618"/>
    <add key="AdminServerMail" value="smtp.gmail.com"/>
    <add key="AdminMailPort" value="25"/>
    <add key="MailsLocation" value="c:\books_store_emails"/>
  </appSettings>

  <system.web>
    <httpRuntime targetFramework="4.5.1" />
    <compilation debug="true" targetFramework="4.5.1" />

    <authentication mode="Forms">
      <forms loginUrl="~/Account/Login" timeout="2880">
        <credentials passwordFormat="Clear">
          <user name="admin" password="1111" />
        </credentials>
      </forms>
    </authentication>
  </system.web>

```



```

    </credentials>
  </forms>
</authentication>

<pages>
  <namespaces>
    <add namespace="System.Web.Helpers" />
    <add namespace="System.Web.Mvc" />
    <add namespace="System.Web.Mvc.Ajax" />
    <add namespace="System.Web.Mvc.Html" />
    <add namespace="System.Web.Optimization" />
    <add namespace="System.Web.Routing" />
    <add namespace="System.Web.WebPages" />
  </namespaces>
</pages>
<profile defaultProvider="DefaultProfileProvider">
  <providers>
    <add name="DefaultProfileProvider" type="System.Web.Providers.DefaultProfileProvider, System.Web.Providers, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" connectionStringName="DefaultConnection" applicationName="/" />
  </providers>
</profile>
<membership defaultProvider="DefaultMembershipProvider">
  <providers>
    <add name="DefaultMembershipProvider" type="System.Web.Providers.DefaultMembershipProvider, System.Web.Providers, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" connectionStringName="DefaultConnection" enablePasswordRetrieval="false" enablePasswordReset="true" requiresQuestionAndAnswer="false" requiresUniqueEmail="false" maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10" applicationName="/" />
  </providers>
</membership>
<roleManager defaultProvider="DefaultRoleProvider">
  <providers>
    <add name="DefaultRoleProvider" type="System.Web.Providers.DefaultRoleProvider, System.Web.Providers, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" connectionStringName="DefaultConnection" applicationName="/" />
  </providers>
</roleManager>
<!--
  If you are deploying to a cloud environment that has multiple web server instances,
  you should change session state mode from "InProc" to "Custom". In addition,
  change the connection string named "DefaultConnection" to connect to an instance
  of SQL Server (including SQL Azure and SQL Compact) instead of to SQL Server Express.
-->
<sessionState mode="InProc" customProvider="DefaultSessionProvider">
  <providers>
    <add name="DefaultSessionProvider" type="System.Web.Providers.DefaultSessionStateProvider, System.Web.Providers, Version=1.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" connectionStringName="DefaultConnection" />
  </providers>
</sessionState>
</system.web>

<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />
  <handlers>
    <remove name="ExtensionlessUrlHandler-ISAPI-4.0_32bit" />
    <remove name="ExtensionlessUrlHandler-ISAPI-4.0_64bit" />
    <remove name="ExtensionlessUrlHandler-Integrated-4.0" />
    <add name="ExtensionlessUrlHandler-ISAPI-4.0_32bit" path="*" verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule" scriptProcessor="%windir%\Microsoft.NET\Framework\v4.0.30319\aspnet_isapi.dll" preCondition="classicMode,runtimeVersionv4.0,bitness32" responseBufferLimit="0" />
    <add name="ExtensionlessUrlHandler-ISAPI-4.0_64bit" path="*" verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" modules="IsapiModule" scriptProcessor="%windir%\Microsoft.NET\Framework64\v4.0.30319\aspnet_isapi.dll" preCondition="classicMode,runtimeVersionv4.0,bitness64" responseBufferLimit="0" />
    <add name="ExtensionlessUrlHandler-Integrated-4.0" path="*" verb="GET,HEAD,POST,DEBUG,PUT,DELETE,PATCH,OPTIONS" type="System.Web.Handlers.TransferRequestHandler" preCondition="integratedMode,runtimeVersionv4.0" />
  </handlers>
</system.webServer>

<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Helpers" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-2.0.0.0" newVersion="2.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>
      <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31bf3856ad364e35" />
      <bindingRedirect oldVersion="1.0.0.0-4.0.0.0" newVersion="4.0.0.0" />
    </dependentAssembly>
    <dependentAssembly>

```

```

    <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
    <bindingRedirect oldVersion="1.0.0.0-2.0.0.0" newVersion="2.0.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="EntityFramework" publicKeyToken="b77a5c561934e089" />
    <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
  </dependentAssembly>
  <dependentAssembly>
    <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
    <bindingRedirect oldVersion="0.0.0.0-1.3.0.0" newVersion="1.3.0.0" />
  </dependentAssembly>
</assemblyBinding>
</runtime>

<entityFramework>
  <defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />
  <providers>
    <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer"
  />
  </providers>
</entityFramework>

</configuration>

```