УДК 621.3

И.В. Пономарев, А.Р. Чухало

ОСОБЕННОСТИ РАЗРАБОТКИ САЙТА С ПОМОЩЬЮ ASP.NET MVC FRAMEWORK

Аннотация. Предложена последовательность действий для создания сайта с помощью MVC технологии. В основе архитектуры разрабатываемого продукта используется шаблон послойной архитектуры, который упрощает работу программиста.

Ключевые слова: ASP.NET, FRAMEWORK, многослойная архитектура, MVC, модель, представление, контроллер.

Введение. Будучи альтернативой технологии Web Forms, ASP.NET MVC использует другой подход к вопросу структурирования веб-приложений. Это означает, что не придется иметь дело с ASPX-страницами и элементами управления, обратными запросами или ViewState, а также жизненными циклами сложных событий. Вместо этого определяются контроллеры, действия и представления.

Постановка задачи. Необходимо рассмотреть особенности разработки сайта с помощью ASP.NET MVC FRAMEWORK, которая позволяет решить проблему костного дизайна приложения и сконцентрироваться на разработке конкретного слоя программы.

Основная часть.

Описание ASP.NET MVC FRAMEWORK.

MVC — это фундаментальный паттерн, который нашел применение во многих технологиях, дал развитие новым технологиям и каждый день облегчает жизнь разработчикам [1]. Рассмотрим составные части платформы [2].

Модель.

Под моделью понимается часть программы, содержащая в себе функциональную бизнес-логику приложения. Модель должна быть полностью независима от остальных частей продукта. Модельный слой ничего не должен знать об элементах дизайна, и каким образом он будет отображаться. Таким образом, достигается результат, позво-

 ISSN 1562-9945

ляющий менять представление данных, и то как они отображаются, не трогая саму модель. Модель обладает следующими признаками:

- модель — это бизнес-логика приложения;
- модель обладает знаниями о себе самой и не знает о контроллерах и представлениях;
- для некоторых проектов модель — это просто слой данных (DAO, база данных, XML-файл);
- для других проектов модель — это менеджер базы данных, набор объектов или просто логика приложения.

Представление.

В обязанности представления входит отображение данных полученных от модели. Однако, представление не может напрямую влиять на модель. Можно говорить, что представление обладает доступом «только чтения» данных. Представление обладает следующими признаками:

- в представлении реализуется отображение данных, которые получаются от модели любым способом;
- представление, в некоторых случаях, может иметь код, реализующий некую бизнес-логику.

Примеры представления: HTML-страница, WPF форма, Windows Form.

Контроллер.

Контроллер преобразует действия пользователя (в данном контексте, пользователь – не обязательно человек) во входящие параметры для модели и передает управление в модель:

- загружает переменные окружения (POST/GET переменные, параметры командной строки, URL параметры и т. д.);
- выполняет первичную обработку переменных окружения (проверка типов переменных, их наличие, установка значений по умолчанию и т. д.);
- реализует механизмы контроля за внештатными ситуациями;
- реализует механизмы логирования (не аутентификации, а ведение журналов).

Последовательность разработки сайта. Для начала необходимо разработать спецификации для пользовательской и административной части. Они должны включать в себя описание основных View-

элементов и соответствующий функционал, который может быть выполнен. В основе архитектуры продукта используется шаблон послойной архитектуры (View Model). При этом разработанное программное решение состоит из пяти слоев, каждый из которых не имеет жестких связей с остальными.

1. Слой сервер, который содержит в себе интерфейсы пользовательских и административных форм, их контроллеры.

2. Репозиторий, которых содержит в себе интерфейс с набором общих CRUD операций и класс с их реализацией.

3. DAL-слой, в котором осуществляется создание базы, посредством Entity Framework.

4. Слой бизнес-логики, который реализует основной функционал для слоя моделей.

5. Слой моделей, которые являются промежуточным слоем между базой данных и бизнес-логикой.

Пример реализации.

В качестве платформы для реализации выбран .Net Framework – мощный, современный, удобный в использовании, инструмент разработки прикладных программ, предоставляющий обширную стандартную библиотеку.

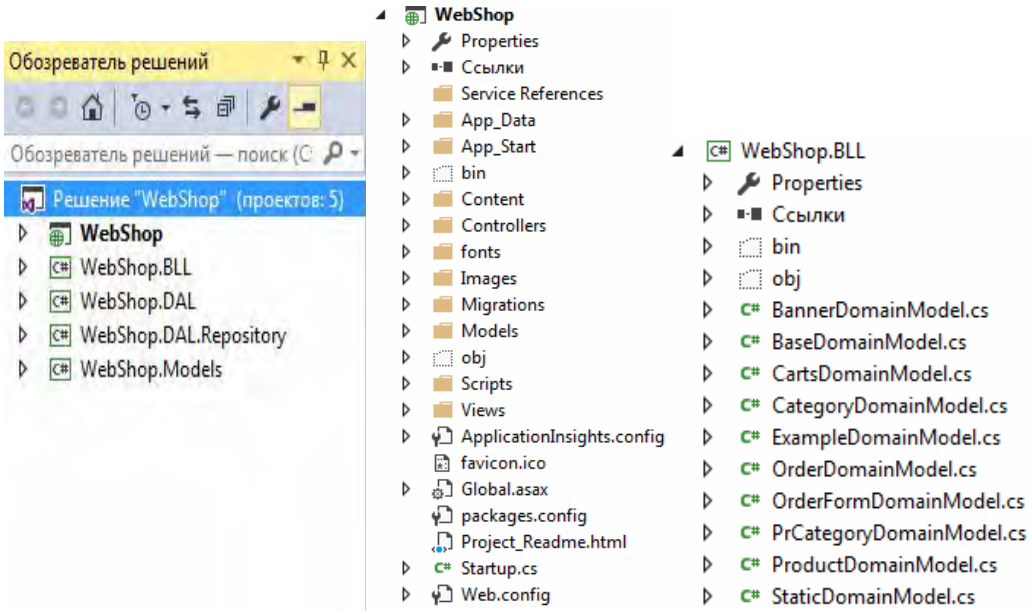

Рисунок 1 - Диаграмма решения и основных слоев сайта

В качестве примера реализации разработан сайт по продаже электроники «WebShop» (рис.1). Рассмотрим статические страницы сайта «WebShop».

 ISSN 1562-9945

```
Таблицы в базе: public class ShopEntities: DbContext
 {
public DbSet<Client> Clients { get; set; }
public DbSet<ClState> ClStates { get; set; }
public DbSet<Image> Images { get; set; }
public DbSet<Order> Orders { get; set; }
public DbSet<OrderState> OrderStates { get; set; }
public DbSet<Product> Products { get; set; }
public DbSet<ProductCategory> ProductCategories { get; set; }
public DbSet<ProductOrder> ProductOrders { get; set; }
public DbSet<PrCategory> PrCategories { get; set; }
public DbSet<Static> Statics { get; set; }
public DbSet<Cart> Carts { get; set; }
public DbSet<OrderCart> OrderCarts { get; set; }
public DbSet<Comment> Comments { get; set; }
}
```

Модель данной таблицы:

```
public class StaticModel : ViewModelBase<int>
   {
          public Int64 Static_id { get; set; }
          public string About_shop { get; set; }
     public string Sales { get; set; }
          public string Delivery { get; set; }
   }
```

Бизнес-логика:

```
public class StaticDomainModel : BaseDomainModel
   {
      public void AddStatic( )
      {
         using (var repository = new BaseRepository<Static,
int>( ))
         {
            //repository.Insert(new Static { Title = "Test1" });
         }
      }
      public IEnumerable<StaticModel> GetAll( )
      {
         using (var repository = new BaseRepository<Static,
int>( ))
         {
            var list = repository.Query( ).Select(x => new
StaticModel
{ Static_id = x.Static_id, About_shop = x.About_shop,
Delivery = x.Delivery, Sales = x.Sales }).ToList( );
```

```
                return list;
            }
        }
        public Static GetById(int id)
        {
            using (var repository = new BaseRepository<Static,
int>())
            {
                return repository.Get(id);
            }
        }
```

Пример страницы «Скидки»:

```
@model IEnumerable<WebShop.Models.StaticModel>
@{
    ViewBag.Title = "Акции";
}
<h2>@ViewBag.Title</h2>
<p>
    @foreach (var item in Model)
            {
        <tr>
            @Html.DisplayFor(model => item.Sales)
        </tr>
    }</p>
```

Выводы. В работе рассмотрена основа программного решения создания сайта с помощью технологии MVC. Определяется архитектура продукта, разрабатывается общая схема взаимодействия слоев программы. Платформа ASP.NET MVC позволяет создавать программы со слабо связанными слоями. Это дает возможность программировать каждый слой независимо друг от друга, таким образом, облегчая работу разработчиков, позволяя сконцентрироваться на конкретной задаче.

ЛИТЕРАТУРА

1. Мартин Фаулер Шаблоны корпоративных приложений. Пер. с англ. – М.: Издательский дом «Вильямс», 2012. - 544 с.
2. Адам Фримен ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов. Пер. с англ. – М.: «Издательский дом «Вильямс», 2016. - 736 с.